Software Engineering Institute

Using Malware Analysis to Tailor SQUARE for Mobile Platforms

Gregory Paul Alice
Nancy R. Mead (Faculty Adviser)

November 2014

TECHNICAL NOTE
CMU/SEI-2014-TN-018

CERT® Division

http://www.sei.cmu.edu

Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

This report was prepared for the
SEI Administrative Agent
AFLCMC/PZM
20 Schilling Circle, Bldg 1305, 3rd floor
Hanscom AFB, MA 01731-2125

DM-0001679

Table of Contents

List of Figures

List of Tables

Acknowledgments

The authors would like to acknowledge the work of Jose Morales at the CERT® division of the Software Engineering Institute (SEI) for providing examples of malware analysis and for providing feedback on the malware analysis conducted in this project. The authors would also like to acknowledge Professor David Root at Carnegie Mellon University for his participation in the requirements review process to complete the Inspect Requirements step of the SQUARE process. Greg Alice would like to thank Nancy Mead at SEI for her support, advice, direction, and feedback in this research project.

Abstract

As the number of mobile-device software applications has grown, so has the amount of malware targeting them. More than 650,000 pieces of malware now target the Android platform. As mobile malware becomes more sophisticated and begins to approach threat levels seen on PC platforms, software development security practices for mobile applications will need to adopt the security practices for PC applications to reduce consumers' exposure to financial and privacy breaches on mobile platforms. This technical note explores the development of security requirements for the K-9 Mail application, an open source email client for the Android operating system. The project's case study (1) used the Security Quality Requirements Engineering (SQUARE) methodology to develop K-9 Mail's security requirements and (2) used malware analysis to identify new security requirements in a proposed extension to the SQUARE process. This second task analyzed the impacts of DroidCleaner, a piece of Android malware, on the security goals of the K-9 Mail application. Based on the findings, new requirements are created to ensure that similar malware cannot compromise the privacy and confidentiality of email contents.

1 Introduction

When a software specification contains the requirement "The system shall be secure" with no further elaboration, the introduction of security vulnerabilities is virtually guaranteed. Security vulnerabilities lead to higher maintenance costs and greater risk to organizations. Security must be integrated into all phases of the software development lifecycle, yet it is frequently overlooked or under-defined in requirements. Requirements definition is conducted at the start of the lifecycle to establish what the software shall and shall not do. Security requirements must be defined with some detail to break the pattern of adding security on, rather than building security into the product. Adding security to software after the project is delivered drives up maintenance costs and endangers internal or external customer's data and corporate reputation. Establishing patterns for security requirements will enable software development teams to utilize best practices in security requirements.

The Security Quality Requirements Engineering (SQUARE) methodology defines a process for a software development team to effectively elicit security requirements from stakeholders and prioritize security requirements based on the security goals of the project. In Step 1, project stakeholders and the development team agree on definitions that will be used throughout the elicitation process. In Step 2, the stakeholders and the requirements engineer identify assets and security goals for the project. In Step 3, the requirements engineer uses the security goals to develop artifacts for the security requirements definition. In Step 4, the requirements engineer, risk expert, and stakeholders use the artifacts developed in Step 3 to conduct a structured risk assessment to define security risks. In Step 5, the requirements engineer selects requirements elicitation technique(s). In Step 6, the stakeholders and requirements engineer elicit security requirements using the risk assessment and artifacts and produce an initial requirements list. In Step 7, the requirements engineer categorizes the security requirements. In Step 8, stakeholders and the requirements engineer prioritize the requirements using the risk assessment and the proposed requirements list. In Step 9, the inspection team inspects and validates the proposed security requirements [Mead 2005].

1.1 Mobile Security Problem

Mobile software development is a rapidly growing area of focus in software development investments. The capability and functionality of new mobile platforms, combined with the convenience of mobile computing, is driving a trend to bring applications to mobile platforms, such as smartphones. The large and rapidly growing number of mobile devices is transforming the computing landscape to make mobile platforms a more valuable target. As a result, the amount of malware for mobile platforms has grown rapidly.

Smartphones have evolved to where PCs were over a decade ago: a trove of valuable data placed on a device whose operators are not familiar with good security practices. As a result, the number of known Android malware samples has grown from just over 100,000 at the beginning of 2013 to more than 650,000 at the beginning of 2014 [Sophos 2014]. The malware problem is exacerbated by the high value the software development industry places on rapid implementation of mobile applications, as well as the rarity of malware that existed on mobile platforms until recently.

These factors have contributed to uneven application of secure software development practices in mobile application development.

1.2 Current Research into the Use of Malware Analysis for Requirements Engineering

The following section details current efforts to use malware analysis as an input for security requirements engineering methodologies.

1.2.1 Malware Analysis in Security Requirements Engineering

The paper "Using Malware Analysis to Improve Security Requirements on Future Systems" [Mead 2014] discusses the development of security requirements using formalized frameworks, such as SQUARE. However, despite improvements in security requirements and secure coding practices, security exploits still occur in software [Mead 2014]. The introduction of security vulnerabilities in software is traced to two sources: coding flaws and design flaws. Coding flaws are a result of the failure to conform to security requirements, so they are beyond the scope of improving the security requirements themselves. Design flaws, on the other hand, are vulnerabilities in the design of the software that can be exploited by attackers.

Mead and Morales argue that design flaws are the result of missed security requirements [Mead 2014]. When a system's requirements contain no security requirement to implement multifactor authentication or separate, secure channels for administration commands in a system, these features are not designed or built into the system. These missing requirements are difficult to identify in a design review, but they can be identified as a form of lessons-learned by analyzing exploited security flaws and generated malware. Designers can examine common failure patterns to identify missing requirements to be incorporated into future releases of software and future applications.

1.2.2 Vulnerability-Centric Requirements Engineering Framework

The vulnerability-centric requirements engineering framework presented by Elahi, Yu, and Zannone seeks to address the gap between (1) the specification of secure components, trust dependencies, threat management, and attacker goals and behavior and (2) the use of actual exploited vulnerabilities as feedback for the development of better requirements [Elahi 2010]. Mead and Morales also note the need to analyze malware to locate deficiencies in the requirements of existing systems in order to provide lessons for the development of future systems [Mead 2014]. SQUARE defines a method for reliably obtaining complete security requirements without requiring specific models, while the vulnerability-centric requirements engineering framework is more prescriptive in identifying specific models to develop in the process of identifying security requirements.

The framework presented in Elahi, Yu, and Zannone is based primarily on the development and refinement of models to elicit security requirements [Elahi 2010]. The process steps of the framework are as follows:

1. **Identify requirements using an i* model.** This step builds the requirements view (goal model). The requirements view captures stakeholders and system actors together with their soft goals, the tasks to achieve those goals, required resources, and the dependencies among them.

2. **Add vulnerabilities to the i* model and propagate the impacts through the model.** This step builds a vulnerability view. The vulnerabilities view extends the requirements view by adding the vulnerabilities that tasks and resources bring to the system and the impact that their exploitation (or their combination) has on the system.
3. **Add attacks and relate them to vulnerabilities by exploit relations** with the goal of developing an attacker template model. The attacker template view captures the behavior of attackers by representing how attackers can exploit vulnerabilities to compromise the system.
4. **Develop attacker profiles.** The attacker profile view captures individual goals, skills, and behavior of a specific class of attackers based on the attacker template view.
5. **Assess risk using goal model evaluation.** The goal model is tested using the dependencies and relations built into the i* framework of the goal model to determine if goals are at risk as a result of the threats.
6. **Add countermeasures, assess their mitigation capability, and model their impacts.** This step builds a countermeasures view by refining the attacker profiles. The countermeasures view captures the security solutions adopted by actors to protect the system and their impacts on attacks and vulnerabilities.
7. **Analyze countermeasures using goal model evaluation.** The effectiveness of the mitigation is quantified, and the i*-based goal model is evaluated with the new mitigations to determine if the goals can be achieved [Elahi 2010].

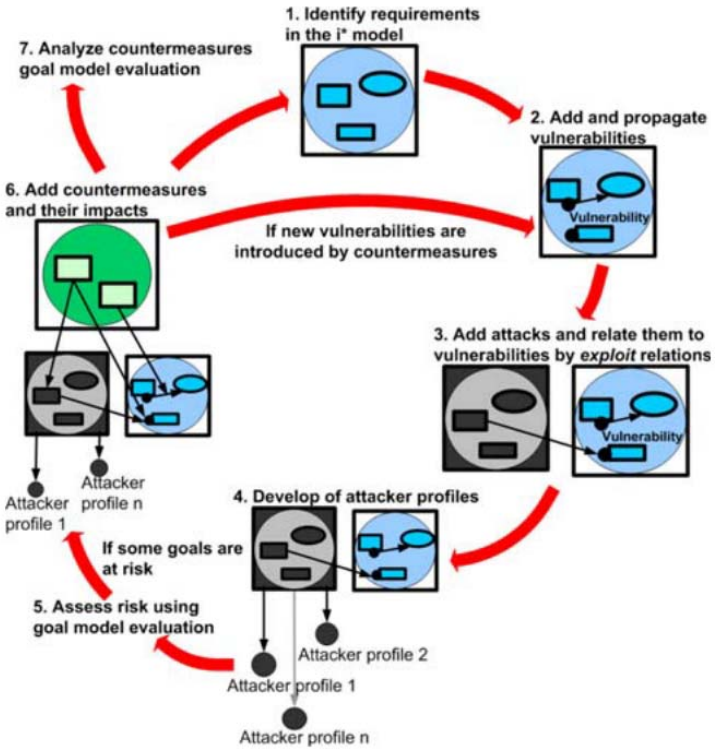

Figure 1: Vulnerability-Centric Requirements Engineering Framework [Elahi 2010]

The vulnerability-centric requirements engineering framework uses vulnerabilities to build attack-tree models and anti-goals. The tree models are used to determine where goals can be defeated

and the system can be compromised. Table 1 maps SQUARE activities to those of the vulnerability-centric requirements engineering framework.

Table 1: Mapping of SQUARE Activities to Vulnerability-Centric Requirements Engineering Framework Activities

SQUARE	Vulnerability-Centric Requirements Engineering Framework
1. Agree on definitions.	*Not defined*
2. Identify assets and security goals.	1. Identify requirements using an i* model – Goal model.
3. Develop artifacts to support security requirements definition.	*Fixed – develop specified models*
4. Perform risk assessment.	2. Add vulnerabilities to the i* model and propagate the impacts through the model. This is not a strong correlation.
5. Select elicitation techniques.	*Fixed - based on model development*
6. Elicit security requirements.	3. Add attacks and relate them to vulnerabilities by exploit relations. 4. Develop attacker profiles. 5. Assess risk using goal model evaluation. 6. Add countermeasures, assess their mitigation capability, and model their impacts.
7. Categorize requirements as to level (system, software, etc.) and whether they are requirements or other kinds of constraints.	*N/A – part of model development*
8. Prioritize requirements.	7. Analyze countermeasures goal model evaluation.
9. Inspect requirements.	*Not defined*

Not all steps in SQUARE have a corresponding activity in the vulnerability-centric requirements engineering framework. SQUARE provides more latitude to select a method of defining requirements, while the vulnerability-centric requirements engineering framework prescribes a process of incrementally refining a threat impact model to arrive at security requirements. The framework does provide a method for collecting models of successful exploits to identify the effectiveness of mitigations. Models of successful exploits need to be developed by a security researcher or malware expert to be effectively defined in the requirements gathering process.

1.2.3 Security Requirements Engineering Process

Mellado, Fernandez-Medina, and Piattini developed the Security Requirements Engineering Process (SREP) [Mellado 2007]. This process utilizes the Common Criteria (ISO/IEC 15408) for security requirements [ISO 2009] and the Systems Security Engineering Capability Maturity Model® (SSE-CMM®) (ISO/IEC 21827) [ISO 2008]. The Unified Software Development Process (Unified Process) is the software development process framework on which SREP is built [Mellado 2007], and SREP activities map to certain phases within the software development lifecycle. SREP supports the reuse of requirements from a common repository; however, it has no defined mechanism for incorporating emerging malware threats from the field. SREP is partially based on SQUARE and shares many of the same process steps [Mellado 2007], as shown in Table 2.

® Capability Maturity Model and CMM are registered marks owned by Carnegie Mellon University.

Table 2: Mapping of SQUARE Activities to SREP Activities

SQUARE	SREP
1. Agree on definitions.	1. Agree on definitions.
2. Identify assets and security goals.	2. Identify vulnerable and/or critical assets. 3. Identify security objectives and dependencies.
3. Develop artifacts to support security requirements definition.	4. Identify threats and develop artifacts.
4. Perform risk assessment.	5. Risk assessment.
5. Select elicitation techniques.	*Not defined*
6. Elicit security requirements.	6. Elicit security requirements.
7. Categorize requirements as to level (system, software, etc.) and whether they are requirements or other kinds of constraints.	7. Categorize and prioritize requirements.
8. Prioritize requirements.	7. Categorize and prioritize requirements.
9. Inspect requirements.	8. Inspect requirements.
Not Defined	9. Repository improvement.

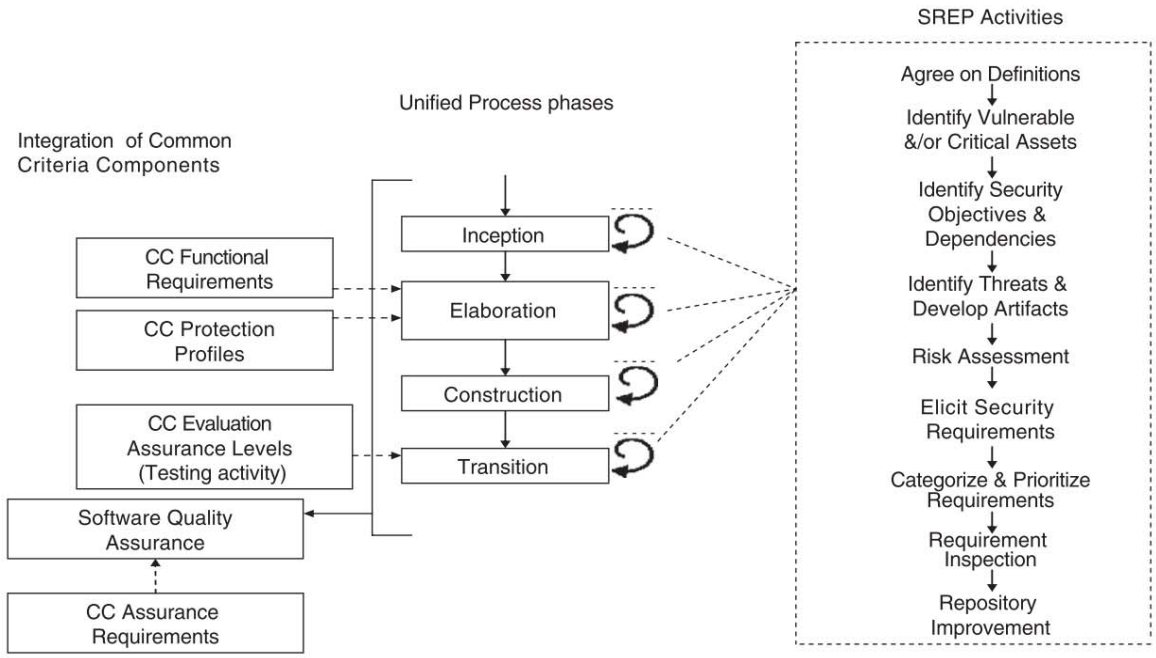

Figure 2: SREP Integration with Unified Process [Mellado 2007]

The Repository Improvement activity is similar to this technical note's proposed extension of the SQUARE methodology: to use malware analysis to identify new security requirements. SREP defines Repository Improvement as the identification of new model elements created as UMLsec models. UMLsec models are software models compliant with the Unified Modeling Language and that are extended to include security aspects. Examples of UMLsec models include misuse cases, threat trees, and attack trees [Mellado 2007]. SREP does not specifically prescribe the method for identifying new security requirements, such as analyzing malware or regularly examining reports from the CERT Division.

1.2.4 Adaptive Security

Salechie and colleagues describe an application that responds to security requirements dynamically. They propose a design pattern for software that dynamically identifies and maps assets, risks, and countermeasures. With the proposed design, software developed using adaptive security will possess a model of its own security posture, which is represented as a cause-and-effect diagram called a causal network. The causal network will contain asset data, security goal parameters, and threat models. New threats or changes to an asset value or goal are fed into the causal network, which causes the system to respond dynamically. Current research envisions the causal network structure obtaining manual updates to the threat model, goal model, or asset model, which would trigger a rebuild of the causal network. The causal network is used to configure an adaptation manager, which controls the response of the software to events. The causal network is a directed graph that contains chance nodes, decision nodes, and utility nodes. Chance nodes represent the likelihood an exploit will occur. Decision nodes represent a mitigation. Utility nodes assess the effectiveness of the mitigations against the exploits [Salechie 2012]. This design is intriguing for its ability to respond to changing risks over the lifetime of software.

Adaptive security does not eliminate the need for security requirements engineering; the proposed design cannot identify new threats on its own and determine how to respond appropriately. Security requirements engineering for such a design requires the development of a matrix of event threat levels and a matrix of appropriate mitigation actions to particular threats and their likelihood of occurring. Adaptive security could improve the usability of software by conditionally activating or disabling intrusive countermeasures. For example, a cell phone operating system (OS) could monitor the phone theft rates in the local geographic area and adjust its screen lockout settings based on the crime statistics, or the feature could be turned on automatically in high-theft areas such as airports.

1.2.5 Malware Ontology

The first step of the SQUARE methodology is “Agree on definitions.” Security requirements require a common vernacular to ensure that all parties involved in the definition of requirements understand them the same way. This common understanding is critical because different stakeholders often have different definitions in mind.

Mundie and McIntire propose the creation of an ontology for malware, which includes common terms, definitions, classifications, categories, and a taxonomy to produce a common vernacular for the understanding of malware [Mundie 2013]. They also advocate for the development of a standard method of describing malware and its attributes to standardize knowledge of malware throughout the malware analysis community.

The malware analysis community needs, in addition to a common vocabulary, a common taxonomy to clarify terms and facilitate malware categorization. An ontology can then be used to define relations between vocabulary entities (objects, concepts, and properties). A malware-specific ontology can be used to build a database that describes all malware, which could be used to identify the attack vectors successfully exploited by malware. Requirements engineers could use this database to identify a reasonably sized set of new requirements that will close common attack vectors. Analytical software could use a formalized malware ontology to identify malware in the wild by

enabling the analytical agent to distinguish malware from the set of all software, an ability that could be used to identify a larger sample of malware for analysis.

From the perspective of requirements engineering, an ontology can be used to classify malware according to its attack vectors and its threats. Based on the assets contained within a proposed software system, requirements engineers could use the malware database to investigate malware that poses the greatest risks. Having identified a set of potential malware entities, their potential attack vectors can be identified and covered by a security requirement.

1.3 Applying Malware Analysis to SQUARE for a Mobile Application

SQUARE can be improved based on the concepts presented in the vulnerability-centric requirements engineering framework, by adopting a method of identifying malware and assessing the security requirements' effectiveness in repelling the threats to the security goals. Step 3 of the SQUARE framework, "Develop artifacts to support requirements elicitation," needs to include malware analysis so that previously successful malware will be considered in the development of security requirements for future systems. Completing malware analysis prior to risk analysis will also allow the current state of malware to influence and enhance the risk analysis process. For example, if SQUARE participants select the "use/misuse cases" technique [Jacobson 1992], the requirements engineering team and the stakeholders should identify potentially impacting malware that will drive misuse cases that threaten assets and security goals. The malware analysis will consist of obtaining the attack vectors used by malware that exploits the given architecture and follows the misuse case pattern. Once the attack vectors are obtained, security requirements are defined to mitigate each identified attack vector. The mitigations will be assessed for effectiveness and cost. The assessment of the mitigations will be presented to the stakeholders during the final phase of requirements elicitation to ensure the cost, performance, and usability of the selected mitigations are acceptable.

The vulnerability-centric requirements engineering framework contains a useful concept for SQUARE: the reassessment of risk after the security requirements are presented. SQUARE could adopt a similar feature by repeating the risk assessment (Step 4) at the end of requirements prioritization (Step 8) to ensure the proposed security requirements have sufficiently reduced the risk of the software system.

2 Survey of Mobile Malware

This section explores the different types of malware present on mobile platforms. Malware was once largely nothing more than petty vandalism, but it has since evolved into a revenue stream for organized crime on the order of the illicit drug trade. It has also become one of the leading mechanisms in industrial espionage [Detica 2011]. This section explains the revenue streams from mobile malware in order to increase understanding of malware authors. Differences between mobile platforms, specifically Google's Android and Apple's iOS, and the subsequent differences in their malware are described. Variations in the two platforms' security models lead to different security requirements and different considerations for application developers. Finally, the mechanics of major types of malware are explained, the attack vectors used by exploits are described, and the consequences of these security deficiencies for system security are elaborated.

2.1 Classes of Mobile Malware

Malware is software with a malicious intent, which can include data disclosure, repudiation, elevation of privileges, and denial of service. Malware subverts users' goals in an effort to benefit the author economically. Categories of malware are similar for PC software and mobile platforms because the advanced feature sets of newer mobile OSs, such as Android, iOS, Windows Phone, and Symbian, provide nearly the same feature sets as computer OSs. Mobile software and cloud computing are maturing technologies that will have a significant presence in future software systems. Mobile software development is growing rapidly and has attracted the attention of criminals. As a result, the number of incidents involving smartphone malware has grown from a few isolated incidents in 2004 to hundreds of thousands of samples [Sophos 2014] in 2014. Mobile malware has rapidly grown in sophistication as malware developers have begun porting malicious attacks from PCs to smartphone platforms. The six categories of malware described below are defined by the following factors:

- method of infection
- method of propagation
- feature set

2.1.1 Trojans

"A Trojan is malware embedded inside of software with a legitimate appearance" [La Polla 2013]. Trojans can have advanced feature sets that utilize the permissions granted to the legitimate-appearing software during installation.

2.1.2 Viruses

"A virus is a self-replicating piece of code that copies itself to another host system through an infected data file or executable" [La Polla 2013]. Viruses spread via the transmission of infected media and typically contain limited feature sets.

2.1.3 Worms

"Worms are programs that automatically replicate themselves through a communication mechanism, such as through an internet connection, network connection, or Bluetooth" [La Polla 2013]. Worms propagate through infected files and executables. Worms typically have limited feature sets.

2.1.4 Rootkits

"Rootkits infect the OS directly" [Peng 2014] and can provide advanced feature sets. Rootkits spread via Trojans and viruses but utilize OS-level functions to evade detection and prevent removal.

2.1.5 Botnets

"Botnets are a class of virus that puts a computing device under the control of a remote host" [Peng 2014]. Botnets are typically developed for organized crime and are used for the purpose of conducting denial-of-service attacks, distributing spam, or obfuscating data transfer [Peng 2014]. Botnets are transmitted in the same manner as viruses.

2.1.6 Hybrid Malware

Hybrid malware uses multiple attack vectors to gain entry to a system. Hybrid malware incorporates elements of several classes of simple malware to create a more advanced feature set for evading detection and collecting data. Hybrid malware such as Oldboot B, which embeds itself into the Android OS, installs malicious software silently in the background, prevents the removal of malware, uninstalls or disables antivirus software, steals data in a manner similar to spyware, and listens to the commands of a central command server such as a botnet. Oldboot B, in particular, uses code obfuscation techniques to evade detection by antivirus software and even has the capability to receive commands through stenography [Dong 2014]. Using a variety of techniques, hybrid malware is able to become more flexible, better evade detection, better prevent removal, spread more rapidly, and collect more data than simple classes of malware.

2.2 Economic Model of Malware

Like many criminal activities, malware writing and distribution is largely motivated by profit. According to data from the first quarter of 2014, 88% of recent mobile malware is profit motivated [F-Secure 2014]. When mobile phones had limited functionality, there was limited access to data and limited financial motive to exploit mobile phones. Mobile phones have since evolved into smartphones, which have the processing capability of a computer from five years ago, a nearly full-featured OS, and additional sensors. With web capabilities, smartphones are used for financial transactions and mobile banking and can even act as a digital wallet. Because smartphones are nearly always on their owner's person, it is a convenient repository for contacts, documents, and financial information.

The earliest smartphone malware profited by tricking smartphones into sending premium-rate SMS messages [Microsoft 2011]. Criminals would set up short message service (SMS) numbers that are premium rated (much like 1-900 phone numbers). They would then propagate malware that caused phones to secretly send messages to these numbers, charging the user for the text mes-

sage and paying the owner of the number. In the beginning of 2014, nearly 83% of all Android Trojans sent SMS messages [F-Secure 2014]. Newer malware is capable of stealing the phone identification data (SIM card and International Mobile Station Equipment Identity Number [IMEI]) to duplicate the phone in another device so that it can snoop on conversations, rack up phone and data use charges, and use premium SMS messages.

Smartphones are highly capable tracking tools. An attacker with remote access to a smartphone can determine the phone's location, operate the camera and audio, and monitor call logs and SMS messages. This tracking capability, on its own, does not have significant monetary value for the average user, but tracking software is being sold to attackers on the internet for the purposes of stalking and other illegal activity [Symantec 2010]. One can envision future "ransomware" that threatens to publish embarrassing or compromising locations tracked on the phone. Smartphone malware could become a serious threat to privacy.

Data theft is another profit stream for mobile malware. Data theft on mobile devices involves the compromise of emails, text messages, call history, contacts, and credentials for applications. Key logging software can be used to steal passwords for financial websites and credit card information [Sophos 2014]. Data theft, particularly of financial credentials, will grow as more financial transactions and purchases occur via smartphone. New pay-by-cell phone options such as Google Wallet will increase the value of smartphone platforms as an avenue for financial theft.

Botnets of smartphones are a potential resource in the distribution of computing platform malware because smartphones are operational and connected to the internet nearly all the time. For instance, existing botnet software already directs infected smartphones to perform repeated web searches to raise the ranking of targeted links in search engine results. As a result of search result manipulation, 1.3% of all search queries result in a malicious link [Paturi 2013]. Search result manipulation can be used as a source of revenue or a method of moving a link to other malware to the top of search results.

Individuals may develop malware to collect revenue in one of the methods described above, or they may sell malware to a larger organization. The potential for significant illicit revenue drives investment in malware by entities such as organized crime [Detica 2011].

Malware that collects documents is used for military or economic espionage. China's efforts at economic espionage are an example of advanced persistent threat (APT). The first APT attack on the Android platform, known as Luckycat, targeted Tibetan activists and specific Japanese organizations in the aerospace and energy industries [Yin 2012]. Other nations' espionage efforts target the data that business professionals and government leaders put onto smartphones.

2.3 Mobile Platforms

The two smartphone platforms with the largest market share in the personal market space are Google's Android and Apple's iOS. The Android platform emphasizes openness, and iOS emphasizes a seamless user experience. As a result of these design philosophies, Google provides a large number of application programming interfaces (APIs) for the Android OS for application developers and an open marketplace called Google Play, where developers publish applications for consumption. Apple provides a limited set of APIs and provides the iTunes store as the only avenue to install new software. All software submitted by developers to iTunes must be vetted by a

proprietary process by Apple prior to distribution. These philosophical differences have strong implications on the security outcome of each platform. Due, in part, to the openness of distribution mechanisms, Android has far more available applications than iOS.

On smartphones platforms, the largest vector of malware by an overwhelming margin is applications. As a result of the open nature of the Android application marketplace, Android has far more malware than other smartphone platforms [F-Secure 2014]. As a result of the APIs and application message broadcasting (called intents), Android provides greater capability for application functionality but also greater potential for harm as a result of malicious code.

Applications published on Google Play are signed by their developers using a certificate. However, Android does not provide for a root-level certificate authority [Khan 2012]. As a result, anyone can write software and publish it under a personal certificate, which gives consumers very little information with which to ascertain the developer's trustworthiness. Android users are largely responsible for assessing the trustworthiness of the product in the market. Google does perform security checks on some applications in Google Play [Ahmad 2013] and will remove applications known to be harmful.

Applications are assigned permissions at the time of installation as requested in the permission manifest file. In other words, at installation time, the application will request access to the services (e.g., internet, SMS, camera, GPS) it will need at runtime. The user must grant access or the application will not be installed [Khan 2012]. This permissions mechanism makes software easier to implement because the application will never have a permission denied error. However, this mechanism forces users to fully trust the software in order to install it. Rather than scrutinize the requested permissions, users are more likely to grant access quickly to install the application. Khan, Nauman, Othman, and Musa present a potential security risk scenario in which two applications signed by the same certificate can share the same process ID [Khan 2012]. When this occurs, the applications can share permissions. A malware author can make a pair of complementary Trojan applications that can each request a separate, innocuous set of permissions. If both applications are installed, their shared process ID allows the two applications to share the union of permissions requested by both. This additive permissions scheme can have significant security implications.

Despite the vastly higher number of exploits on the Android platform, the security posture of Android is not any less inherently secure than iOS. Android sandboxes individual applications, while iOS has all applications within the same sandbox. If web browser attacks were targeted at browsers available on iOS, an exploit could escape the web browser and put application data at risk. Security software is also available for Android but not for iOS [Ahmad 2013]. Apple does not publish iOS vulnerability statistics, but because no developers are infallible it can be assumed that security issues exist in iOS, just as in any other OS. However, considering that the largest attack vector for smartphone platforms is the application layer, Apple's walled garden approach to the application marketplace does provide a significant security advantage.

3 Case Study Results

In their paper "Using Malware Analysis to Improve Security Requirements on Future Systems," the authors proposed adjusting the SQUARE process to account for the implications of mobile malware, specifically by modifying the SQUARE methodology to include malware analysis in Step 3, "Develop Artifacts to Support Requirements Elicitation." The following case study was conducted to assess the effectiveness of this proposed extension through the use of the SQUARE methodology in the development of security requirements for a mobile application, K-9 Mail.

3.1 Project Application

3.1.1 Application Description and Functionality

K-9 Mail is an Android mail client application that is compatible with a variety of email services, such as IMAP, POP3, and Exchange 2003/2007. K-9 Mail is open source with search functionality, IMAP push, multifolder syncing, email flagging, filing, signatures, bcc-self, Pretty Good Privacy (PGP) encryption, and mail on a mobile storage device. When installing the tool, it asks for permission to access Contacts and Calendar, as well as Photos, Media, and Files [K-9 2014].

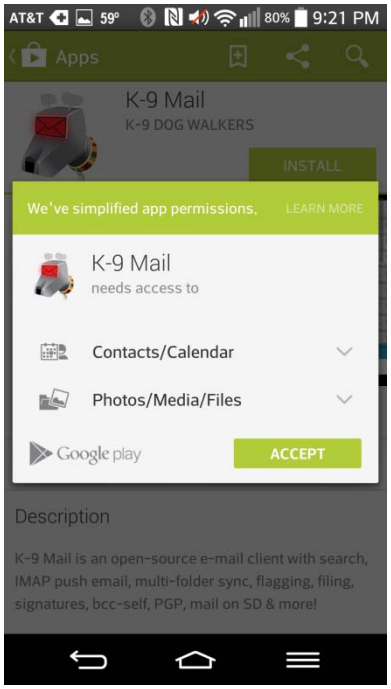

Figure 3: Permissions Request on Install of K-9 Mail

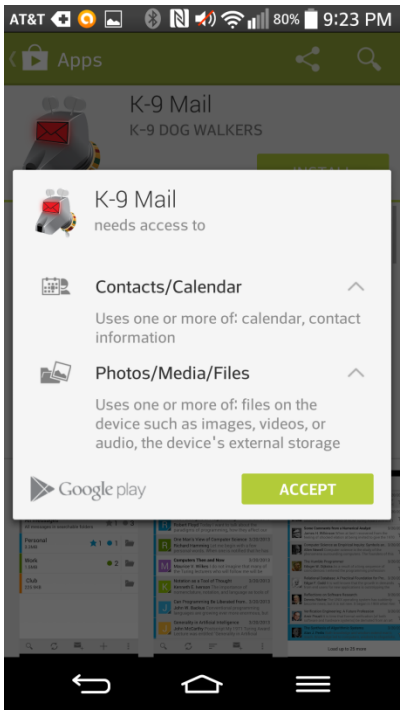

Figure 4: Detailed Permissions Request for K-9 Mail

Once installed, the application has the typical functionality one would expect for an email client on a mobile phone, such as manual syncing, multiple-account support, and email searching.

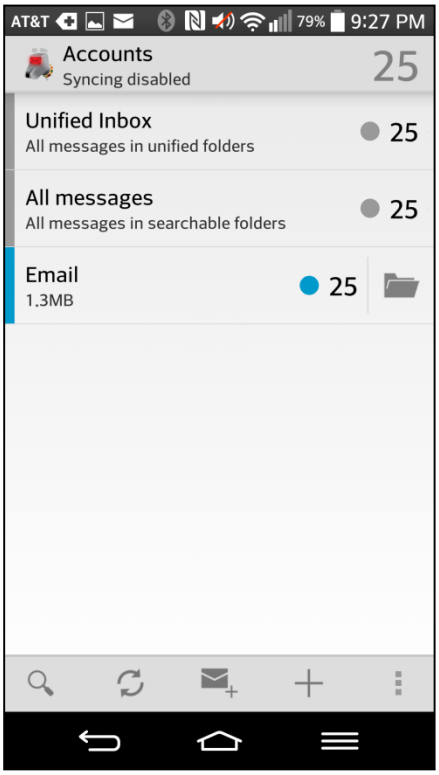

Figure 5: Default K-9 Interface Startup Window

Recent changes to the application include improvements to the encryption of communication between the client and email server.

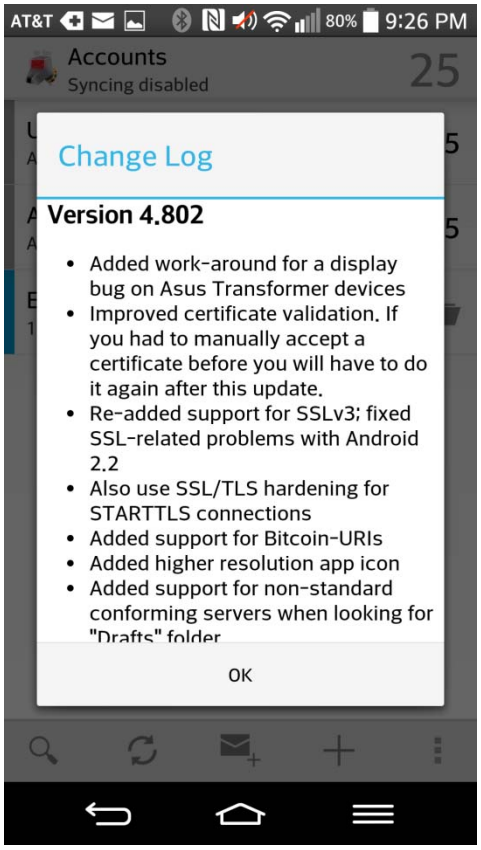

Figure 6: Change Log

3.1.2 Technical Information

K-9 Mail has 32 Java packages, 448 classes, 57,939 lines of code, an average inheritance depth of 2.28, and a maximum cyclomatic complexity value of 486 for the method that sorts out the correct MIME type of attachments. This application is moderately complex by Android application standards, but its code base is not large by software development industry standards.

3.2 SQUARE Step 1: Agree on Definitions

Especially in highly technical domains of knowledge, agreeing on definitions is critical for effective communication. Software development requirements represent a contract of complex terms between the customer organization and the software development organization, so using agreed-upon definitions that employ vernacular common to both parties is vital.

This technical note uses the following definitions for the security requirements of the K-9 Mail application:

- malware—software written with a malicious intent designed to compromise the security goals of the project [La Polla 2013]

- Trojan—software, deployed to an application store, that is designed to appear to be legitimate but contains embedded malware [La Polla 2013]
- spyware—software designed to compromise the confidentiality of data belonging to an individual or organization without the full consent of the observed parties [Microsoft 2014]
- vulnerability—a weakness in use of the application that increases the potential for harm by a damaging event [Elahi 2010]. The damaging event may be caused by malfeasance, user action regardless of intent, unintended consequence, or act of nature. Vulnerabilities can occur in user actions, processes of use, or within the software system itself.
- exploit—an intentional event that causes a vulnerability to be realized as a loss [Elahi 2010]
- client—the Android-based software application that connects to the email server and performs actions on the server through a connection
- server—the email server that responds to requests for email, contains the inbox, and manages credentials

3.3 SQUARE Step 2: Identify Assets and Security Goals

To examine the security threats to an application, the application's security goals must be understood. A calculator application has very different security requirements than an online banking application. The SQUARE process places security goals near the beginning of the process, immediately following definitions [Mead 2005]. Email is used for personal communication and business communication. Emails may contain inconsequential messages or they may contain sensitive data, such as banking information, PII, medical information, account information, or links into password reset tools. Due to potentially sensitive data transmitted via email, users have an expectation of privacy and security commensurate to the value of the data contained within email. The confidentiality and availability of email is also important to users. Because email inherently involves client-server interaction and local storage, data at rest and data in transit must be considered.

The security goals of an email client are the following:

- The email account credentials shall be protected.
- Email in transit between the server and the client shall be protected.
- Email stored on the client shall be protected from unauthorized access.
- Email shall not be sent without permission of the account owner.
- Attachments stored on the client shall be protected from unauthorized access.

3.4 SQUARE Step 3: Develop Artifacts to Support Security Requirements Definition

3.4.1 Android Security Model

Android is derived from the Linux kernel and inherits many of its security features. Application sandboxing in Android is achieved by running each application as a separate user in the underlying Linux kernel [Google 2014]. This design choice leverages Linux's ability to separate application data, provide secure interprocess communications, and isolate processes. Most Android OS

functions operate at application-level permissions as well [Google 2014], so cracking OS libraries will not give an attacker a foothold into other applications.

Android provides each application with an application data storage area called Internal Storage, which is controlled by the application's ID [Google 2014] much in the way a user in UNIX has control over the personal Home directory. Android also has a general storage area called External Storage, which may be in the form of removable media or an emulated storage area in the device's file system. In contrast to the stringent controls on Internal Storage, the External Storage area is granted permissions at the top level [Ahmad 2013]; that is, if read access is required to a single directory in External Storage, read access must be granted to the entire storage area.

As described in Section 2.3, users often grant applications' requested permissions without scrutinizing them, which gives malware the permissions it needs to wreak havoc.

There are three primary methods by which Android's robust security model will fail to protect the application data of K-9 Mail:

1. The application mistakenly grants broad access to the application's data storage area.
2. The application stores data outside of its application storage area, such as in External Storage.
3. A process other than the Linux kernel obtains root privileges. This can occur when a user "roots" his or her device or when a Trojan contains an attack against the OS. Malware designed to obtain root privileges typically uses the same code available for ambitious Android users who wish to root their devices.

K-9 Mail does not grant broad access to its application storage area, so it does not exhibit the first vulnerability above. K-9 Mail does allow users to save email data in the External Storage area, so it does exhibit the second vulnerability. The third vulnerability occurs when the entire Android security model has been compromised. There is very little the K-9 Mail application can do about this vulnerability, just as fire-resistant fabric will not hold up in a house on fire.

DroidCleaner is a piece of Android malware that can defeat K-9 Mail's security model when the application is configured to store email in External Storage. Because External Storage has drive-level permissions, DroidCleaner is able to ask the user for READ_EXTERNAL_STORAGE and INTERNET permissions on installation [Chebyshev 2013]. When permissions are granted to READ_EXTERNAL_STORAGE, DroidCleaner has access to all the data stored in this storage area. INTERNET permissions open up network access, which gives DroidCleaner a channel for transferring data to the attacker.

3.4.2 Malware Class Analysis

The development of security requirements for mobile applications should include malware analysis. This case study analyzed malware for mobile platforms. Common patterns were identified in the malware's attack vector. The security goals of the application were assessed against the malware's goals to determine if there exist classes of malware that pose a threat to the realization of the application security goals. The following classes of malware are identified for mobile platforms.

3.4.2.1 Spyware

Spyware compromises the confidentiality of a system and targets a user's privacy. Spyware typically enters a smartphone through a Trojan [Sophos 2014]. It then proceeds to access data on the phone and sends the data through the internet to the entity who employed the spyware. Depending on the variant, spyware can

- monitor all activity on the phone (calls and SMS)
- collect email
- turn on the microphone and camera (to spy)
- report location
- collect stored credentials (to impersonate the user on social media and on financial websites)
- log keystrokes (to collect passwords)

Spyware collects data and sends it to the author, or it opens a backdoor in the device for later exploitation.

3.4.2.2 SMS Messaging

SMS messaging attacks are typically introduced through Trojans, though botnets may be employed to generate revenue more rapidly. Section 2.2 describes the economic model behind SMS messaging attacks. The earliest smartphone malware performed SMS messaging and is still by far the most common type of attack carried out on smartphones. To evade detection, advanced SMS messaging malware will intercept incoming text messages from mobile phone carriers to prevent the user from being notified of the fees incurred. Other pieces of malware employing SMS messaging are implemented as rootkits to make the process generating the SMS messages invisible to the user. Rootkit SMS messaging malware is designed to reinstall itself if it is removed [Jiang 2011].

3.4.2.3 SIM Theft

Certain smartphone malware will access the phone's IMEI and International Mobile Subscriber Identity (IMSI) number and send it back to the malware author. The IMEI is built into the device and uniquely identifies it, and the IMSI (from the SIM card) uniquely identifies a user on a mobile network. The theft of these two identifiers allows someone to masquerade on the mobile network as a different user, rack up mobile charges, and potentially intercept communications such as SMS messages. SIM theft typically exploits a Trojan as the attack vector [Balanza 2011].

3.4.2.4 Ransomware

Ransomware is software that locks a file, data, or use of the phone until a payment is made to the malware author. Ransomware's attack vector is also a Trojan. Android Defender is an example of Android ransomware. The software used social engineering to request device administration privileges that, once granted, it used to lock out all functionality on the phone and present the victim with a demand for $99.99 to regain access. With administration privileges, Android Defender was capable of disabling all buttons, launching itself on reboot, and performing a factory reset [Walker 2013]. Ransomware is removed from markets as soon as the market operators are made aware of the issue, but there are black market application stores that provide no protection for users.

3.4.2.5 Adware

Adware is software that aggressively pushes advertisements on users and is frequently bundled with free applications. Some variations of adware will intrusively violate the user's privacy to target advertisements, such as by monitoring web browsing and the content of communications. Other adware displays advertisements in the OS's notifications and changes web browser bookmarks. Adware is typically distributed through Trojans [Symantec 2012].

There is controversy within the security community about whether adware is truly a form of malware. In 2012, an Android software development toolkit named Apperhand was introduced that had aggressive data collection policies. Symantec classified it as malware, but mobile security company Lookout decided against labeling it as malware and did not include it in the company's antivirus definitions [Lookout 2012]. Ultimately, the classification of adware as malware versus legitimate advertising comes down to respect for users' rights: if users agree to certain levels of data collection in exchange for free software, and the software only collects the mutually agreed-on data, then adware is permissible. Privacy policies for adware should clearly state what data is collected and with whom the data will be shared. Also, legitimate adware should stop collecting data and displaying advertisements on removal of the software.

3.4.2.6 Data Stealing

Data stealing is an attack that retrieves confidential data from a user's device. The attack vector for data stealing on mobile platforms is primarily Trojans, though rootkits and botnets can also be used. Typically, in this type of attack, an exploit is used to open a backdoor into the smartphone. A typical data-stealing attack will follow these steps:

1. A Trojan is placed on a smartphone platform's application store.
2. The victim installs the software containing the Trojan on a smartphone.
3. The Trojan uses its permissions to make changes to the smartphone's OS to gain access to files.
4. The Trojan opens the storage on the device to the internet.
5. The Trojan informs the malware author of the smartphone that is now under the author's control.
6. The malware author uses the permissions granted by the Trojan to access the smartphone's storage to copy files and data stored on it.

Luckycat is an example of data-stealing malware that was implemented using a Trojan [Yin 2012]. Data stealing targets usernames and passwords for banking sites, contact information, and documents.

3.4.2.7 Botnets

Botnet attacks are typically conducted via the vector of a Trojan application. Once an application containing a Trojan is installed, the software begins monitoring for commands sent from the command and control system. Botnet attacks can be thwarted by blocking the command and control server. However, new malware is designed to dynamically change command and control server. A few new pieces of malware even receive their commands through an encrypted channel via a network of thousands of relays, so the web traffic is virtually impossible to track [Unuchek 2014]. Because the source of the attack cannot be determined, internet service providers are una-

ble to disconnect command and control servers from their networks. Botnets are typically used for distributed denial-of-service attacks and are less frequently used to illegally collect data or spread malware.

3.4.2.8 Denial of Service Attacks

Denial-of-service (DoS) attacks aim to disrupt services, such as access to websites or mobile networks. DoS attacks are frequently implemented using a distributed method in which many different devices coordinate to load the same network or computer node at the same time. Such distributed DoS attacks are typically propagated through botnets of PCs, but this may change with the rapidly growing number of mobile devices. DoS attacks targeting mobile phone networks would be much easier to implement using smartphones than by using other methods. Mobile communications could be disrupted to delay response to a military action or terrorist attack.

3.4.2.9 Summary

Spyware and data stealing are the most likely compromises to interfere with the achievement of security goals for the K-9 Mail application.

3.4.3 Malware Analysis—Application Vulnerability

To identify new security requirements, the DroidCleaner Trojan was examined for its potential compromise of the K-9 Mail application. DroidCleaner was not designed to target the K-9 Mail application directly, but it is a data-stealing application. One of the key features of DroidCleaner is its capability to upload all of the contents of the device's External Storage directories to a remote server under the control of the malware designers [Paoli 2013]. What the malware designers do with the contents of the uploaded data is unknown, but they likely examine it for valuable private information. We believe that future attacks using a similar vector and with a similar purpose—to steal data and extract value from it—are likely as attackers harden the current mobile malware monetization method of using premium-rate SMS messages and as the value of data stored on mobile devices increases.

K-9 Mail can be configured to store data in Internal Storage or the External Storage directories [K-9 2014]. The DroidCleaner malware is capable of uploading the K-9 Mail data stored in the device's External Storage to an attacker's computer. The following section will examine the impacts of such an exploit. The exploit was simulated by changing the settings in the K-9 Mail client for sample email accounts to use External Storage and observing the newly visible directory in <External Storage>/Android/data, copying it to a PC, and examining it. K-9 Mail stores the following data in External Storage (see Figure 7).

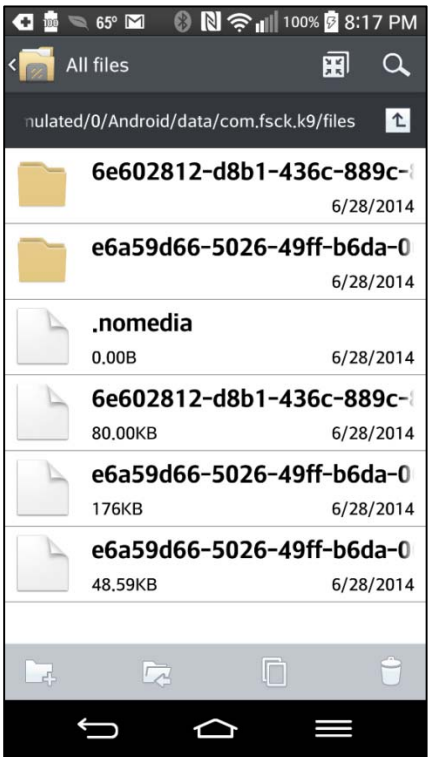

Figure 7: K-9 External Storage Files

The data seems cryptic when viewed on the phone, but each of the bottom three files is an SQLite database corresponding to an email account. Figure 8 shows the contents of one of the directories.

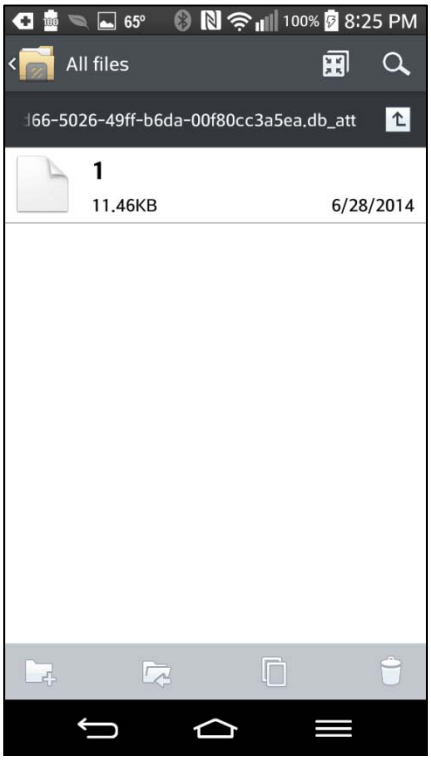

Figure 8: K-9 Mail Attachments

The directory contains a file with no extension, but examination reveals that it corresponds to an opened attachment.

To assess the potential damage from a data-stealing attack, such as by DroidCleaner, the contents of the K-9 Mail data directory were loaded to a PC for examination. The examination of K-9 Mail data files revealed the information shown in Figure 9.

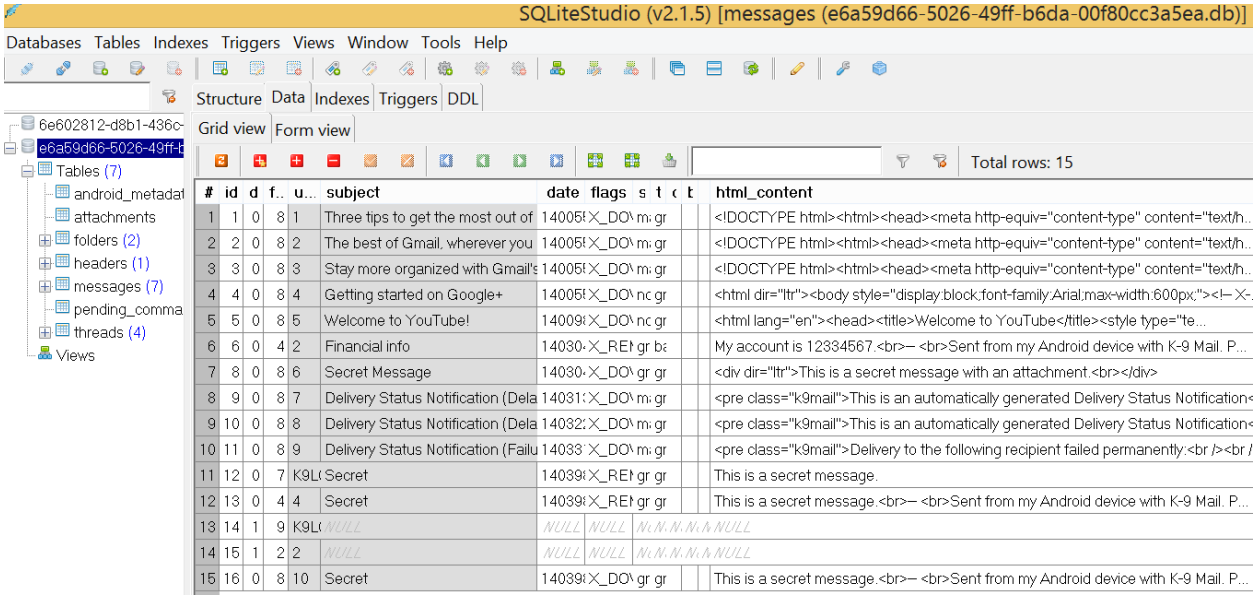

Figure 9: K-9 Mail Database Contents

The SQLite database is not encrypted, so on transferring the file to a PC, the contents of the email account display in clear text. When the extension-less file "1" from the attachment directory is loaded onto a PC and opened in its native application (Word), it is also unencrypted and displays in clear text. The ability of K-9 data files to be copied from Android External Storage to another location by DroidCleaner or other malware and read in clear text represents a confidentiality vulnerability in the application. K-9 Mail users who are victims of DroidCleaner or a similar data-stealing Trojan are at risk of having their email contents exposed.

3.4.3.1 Attack Trees

The attack tree shown in Figure 10 shows an attack tree for the exploit detailed in Section 3.4.3. This attack tree was used to develop the following misuse cases for Step 6 of SQUARE: *MUC1 – Email Credentials are stolen* (see Section 3.7.2.3), *MUC2 – Data in an email stored on the smartphone is stolen* (see Section 3.7.2.1), *MUC4 – Confidential data in attachments is stolen* (see Section 3.7.2.2), *MUC6 – Emails are deleted without user's consent* (see Section 3.7.2.6), and *MUC7 – Attachments are deleted without user's consent* (see Section 3.7.2.7). The attacker's goal is to steal data contained within emails. It is possible to achieve this goal by using an email attachment containing malware or by deploying a Trojan installed by the victim. The K-9 Mail data is compromised when the attacker gains access to the location within the smartphone's storage containing the K-9 Mail files. The attacker may steal data files or steal the stored credentials used by the K-9 Mail application to authenticate itself to the email server.

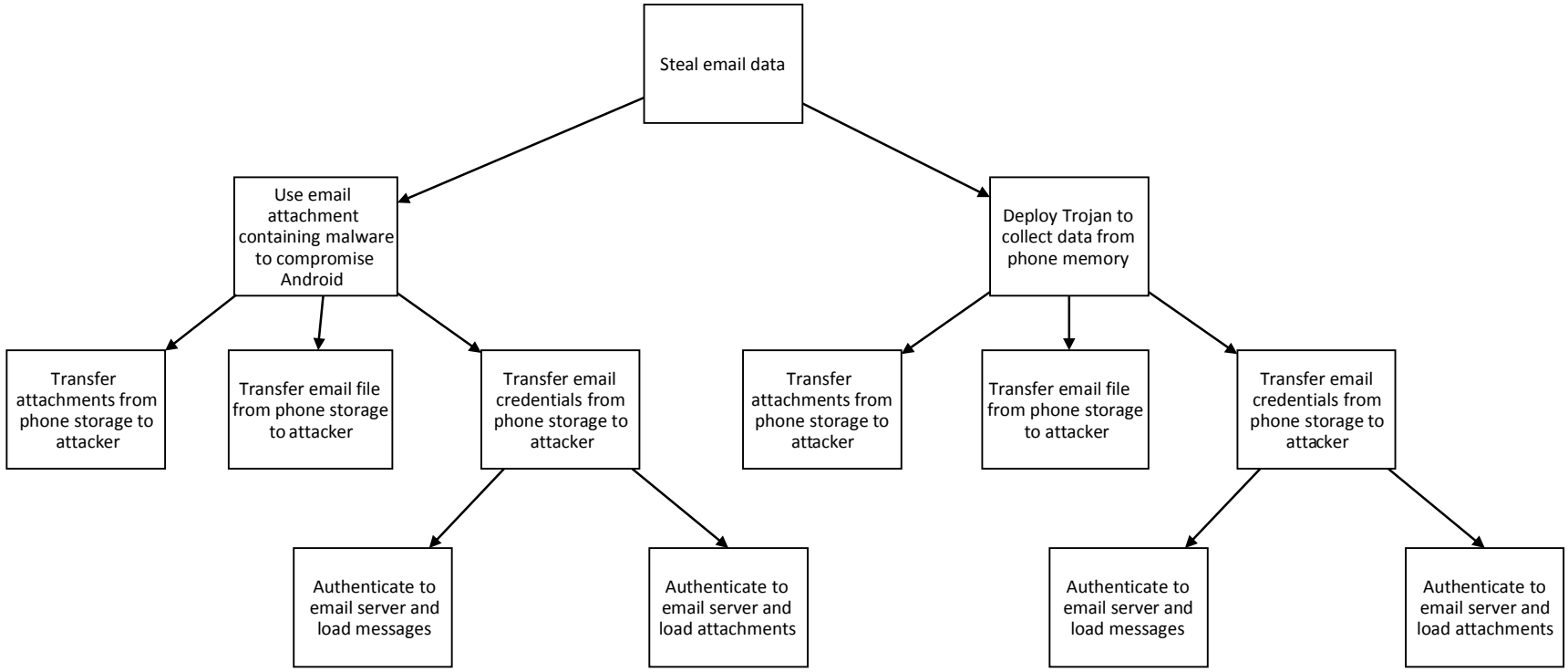

Figure 10: Attack Tree—Data Stealing of Email Content and Attachments Exploited by DroidCleaner

The attack tree shown in Figure 11 describes misuse case *MUC9 – K-9 Mail application is placed on Google Play as a Trojan*. The attacker's goal may be to add the smartphone to a botnet, steal data, or send premium SMS messages. A Trojan may accomplish any of these goals. The attacker may choose to use the default permissions requested by the K-9 Mail application, or the attacker may configure the malware to request additional permissions, create a more involved attack, and gain greater access to data stored on the device.

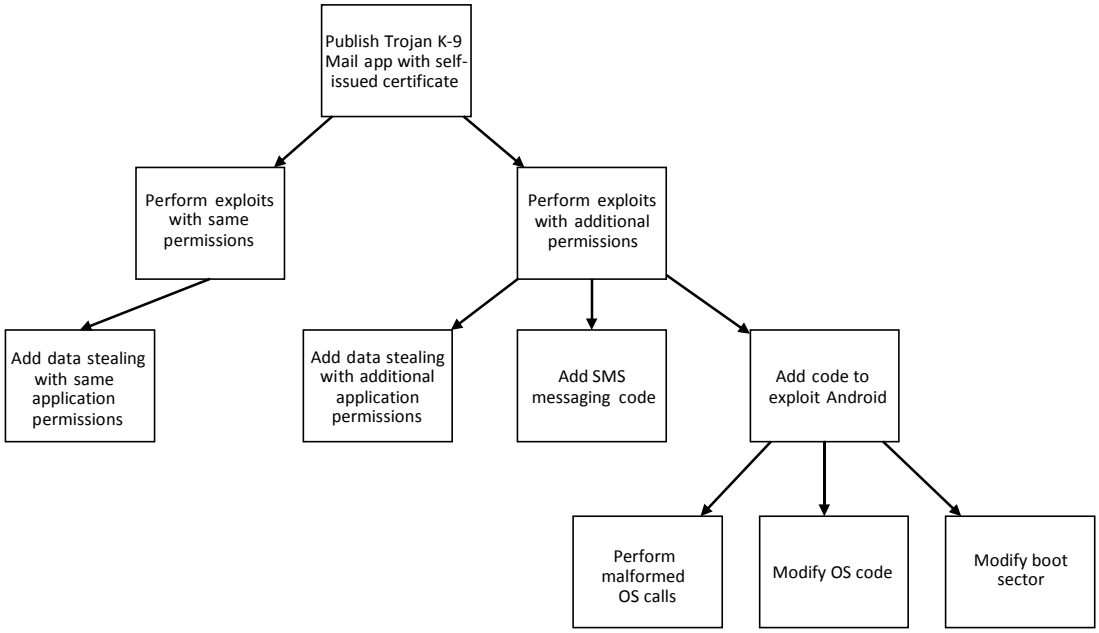

Figure 11: Attack Tree—Trojan K-9 Mail Application

3.5 SQUARE Step 4: Perform Risk Assessment

This case study used the NIST Risk Assessment.

3.5.1 Risk Identification

The first step of the NIST Risk Assessment process is to identify threat sources and then vulnerabilities. Based on the previously identified security goals, the risks shown in Table 3 were identified from the malware classes in Section 2.1. DroidCleaner exploits Risks 2 and 4. The remainder of the risks were identified by examining the malware survey and determining which existing classes of malware, with some modifications, could pose a threat to the security goals of the K-9 Mail application.

Table 3: Malware Risks to K-9 Mail

ID	Risk
1	Email credentials are stolen.
2	Data in an email stored on the smartphone is stolen.
3	Data in an email in transit is stolen.
4	Confidential data in attachments is stolen.
5	Attachment contains malware and is used to compromise the device.
6	Emails are deleted without user's consent.
7	Attachments are deleted without user's consent.
8	Emails are sent without user's consent.
9	K-9 Mail application is placed on Google Play as a Trojan.

3.5.2 Malware and Risk Matrix

Table 4 shows the matrix of risk IDs mapped to different classes of mobile malware. The potential malware entities are assessed against the risks based on the feature sets inherent in the malware from the mobile malware survey (see Section 3.4.2).

Table 4: Mapping of Malware to Risk ID

Malware	Risk								
	1	2	3	4	5	6	7	8	9
Spyware	X	X	X	X					X
SMS messaging					X				X
SIM theft					X				X
Ransomware						X	X		
Adware								X	
Data stealing	X	X	X	X					X
Botnets					X				X
Denial of service									

Based on the matrix, spyware and data-stealing attacks have the potential to trigger events for the greatest number of risks.

3.5.3 Malware Risk Assessment

3.5.3.1 DroidCleaner

The Common Vulnerability Scoring System (CVSS)[1] was used to determine the impact of the vulnerability of K-9 Mail to a data-stealing attack such as DroidCleaner. This type of attack falls into the Common Weakness Enumeration's[2] issue 921: "Storage of Sensitive Data in a Mechanism without Access Control" [MITRE 2014]. CVSS was used to determine a risk score, shown

[1] CVSS is a vulnerability scoring system designed to provide an open and standardized method for rating IT vulnerabilities. It was created by the Forum of Incident Response and Security Teams (FIRST) and the Common Vulnerability Scoring System-Special Interest Group (CVSS-SIG). See http://www.first.org/cvss.

[2] The Common Weakness Enumeration (CWE) is a community-developed dictionary of software weakness types maintained by MITRE Corporation. See http://cwe.mitre.org/.

in Figure 12, for this vulnerability using the criteria provided in the tool to select appropriate values. The criteria are shown in Figure 13.

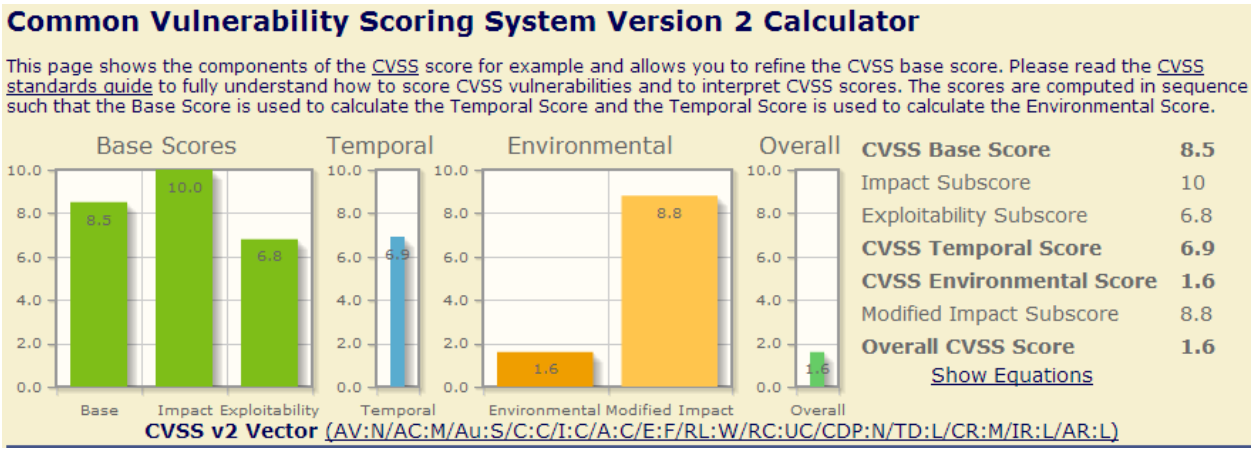

Figure 12: CVSS Score for a DroidCleaner Attack Against K-9 Mail [NIST 2014]

Figure 13: Selections for DroidCleaner Exploit in CVSS [NIST 2014]

The high severity of the attack is offset by the failure to confirm that DroidCleaner extracted highly confidential data from email clients. Furthermore, a workaround exists for this vulnerability: not storing email in the External Storage area of Android.

3.5.3.2 All Malware

The malware risk assessment for the elicitation of security requirements was conducted by examining the malware classes and selecting the most appropriate values in the CVSS survey, based on the malware observed in the malware survey in Section 2. Overall, CVSS scores for all risks are lower than scores for the component risks because very little of the malware observed in the wild attacks the application level of Android systems; the vast majority use social engineering tactics to access the SMS messaging functionality and send premium-rate SMS messages. Furthermore, while many exploits have a high impact, their likelihood (exploitability score) is quite low due to mitigating factors in the implementation of K-9 Mail, such as storing credentials in an encrypted file in a secure area of the Android file system. The observable incidence of malware targeting the application layer is very low for mobile platforms, so the CVSS tool adjusts the overall score downward based on the low temporal score metrics given to the tool.

Table 5: CVSS Scores for K-9 Mail Malware Risks [NIST 2014]

ID	Risk	Base	Impact	Exploitability	Overall CVSS Score	CVSS Vector
1	Email credentials are stolen.	6.2	9.2	3.2	1.3	(AV:N/AC:M/Au:S/C:C/I:C/A:N/E:U/RL:ND/RC:UC/CDP:L/TD:L/CR:H/IR:H/AR:H)
2	Data in an email stored on the smartphone is stolen.	8.5	10	6.8	1.6	Above
3	Data in an email in transit is stolen.	2.7	4.9	2	0.9	(AV:A/AC:H/Au:M/C:P/I:P/A:N/E:U/RL:OF/RC:UC/CDP:N/TD:N/CR:H/IR:H/AR:H)
4	Confidential data in attachments is stolen.	8.5	10	6.8	1.6	Same as above
5	Attachment contains malware and is used to compromise the device.	6.5	6.4	8	1.5	(AV:N/AC:L/Au:S/C:P/I:P/A:P/E:POC/RL:OF/RC:UC/CDP:L/TD:L/CR:H/IR:H/AR:H)
6	Emails are deleted without user's consent.	5.7	8.5	3.2	1.3	(AV:N/AC:H/Au:M/C:P/I:P/A:C/E:U/RL:OF/RC:UC/CDP:L/TD:L/CR:H/IR:H/AR:H)
7	Attachments are deleted without user's consent.	5.7	8.5	3.2	1.3	(AV:N/AC:H/Au:M/C:P/I:P/A:C/E:U/RL:OF/RC:UC/CDP:L/TD:L/CR:H/IR:H/AR:H)
8	Emails are sent without user's consent.	2.5	7.8	3.2	1.1	(AV:N/AC:H/Au:M/C:P/I:C/A:N/E:U/RL:OF/RC:UC/CDP:N/TD:L/CR:H/IR:H/AR:H)
9	K-9 Mail application is placed on Google Play as a Trojan.	7.9	9.2	6.8	1.8	(AV:N/AC:M/Au:S/C:C/I:C/A:N/E:U/RL:W/RC:UC/CDP:LM/TD:L/CR:H/IR:H/AR:H)

Based on the scores from CVSS, the biggest risks to the K-9 project are

- Risk 9: K-9 Mail application is placed on Google Play as a Trojan.
- Risk 2: Data in an email stored on the smartphone is stolen.

- Risk 4: Confidential data in attachments is stolen.
- Risk 5: Attachment contains malware and is used to compromise the device.

The Trojan risk is large due to the frequency of Trojan exploits on the Android platform. However, the Trojan risk is endemic to Android rather than inherent in an email client. The remaining risks are quite small due to mitigating factors, such as encryption or the application protection using the Android security model.

3.6 SQUARE Step 5: Select Elicitation Techniques

Due to the small size of the project and the minimal functional requirements documentation of the K-9 Mail application, the requirements elicitation techniques should capture all of the security requirements with a minimal amount of process overhead. The selected elicitation technique was used in conjunction with the data obtained from malware analysis. The Accelerated Requirements Method (ARM) was selected as the requirements elicitation technique from the suggested methods in the technical report *Security Quality Requirements Engineering (SQUARE) Methodology* [Mead 2005]. ARM is a quick, scalable method that involves Brainstorm, Organize, and Name activities. In this technique, a focus question is presented to an expert who quickly identifies security requirements. After the initial security requirements definition, the requirements are collected, categorized, and refined.

3.7 SQUARE Step 6: Elicit Security Requirements

3.7.1 Accelerated Requirements Method

ARM identifies a focus question that elicits ways in which the security goals could be compromised. The answers to the focus question were collected and used to brainstorm an initial requirements list. This case study extended the process to create additional requirements, by developing misuse cases based on identified threats to the security goals.

3.7.1.1 Focus Question

The focus question selected for the accelerated requirements method was “Is the confidentiality of email threatened when _____?” For the purposes of requirements elicitation, the blank is an event in the use of the application. The event was then used to derive security goals.

3.7.1.2 Focus Question Responses

Response 1: email is sent

Derived security goals:

- Data must be protected while in transit.
- Credentials stored on the phone must be protected.

Response 2: the phone is in use

Derived security goals:

- Email contents stored on the phone must be protected.

- Downloaded email attachments must be protected.

Response 3: attachments are opened

Derived security goals:

- Attachments must be opened in a manner that will not compromise the security of the Android OS.

Response 4: the application is installed

Derived security goals:

- K-9 Mail must have defenses from Trojan applications.

3.7.1.3 Initial Requirement List

Misuse cases were analyzed to draw out security requirements. The focus question from the ARM method was analyzed, and as many requirements were identified as possible within seven minutes. The results from the elicitation technique, ARM, and the developed artifacts (malware analysis and attack trees) were combined into the following initial requirements list:

1. Email shall be protected from unauthorized access.
2. Attachments shall be protected from unauthorized access.
3. K-9 Mail shall prevent the spread of Trojan versions of the K-9 Mail application code (copies of the compiled application with additional malware code).
   - Note: Because K-9 Mail is open source, it is not possible to restrict access to the code. It is possible for attackers to add additional code to their own branch of the code, compile the application, and distribute. Mechanisms need to be put in place to ensure that users are able to identify imposter versions of K-9 Mail.
4. K-9 Mail shall prevent attachments from compromising the OS.
5. K-9 Mail shall protect email credentials.
6. K-9 Mail shall protect data in transit.

3.7.2 Misuse Cases

To analyze the completeness of the initial requirements list, the following misuse cases were developed for the K-9 Mail application. The first misuse case shown (MUC2) was intentionally executed on the author's account and studied extensively end-to-end, and it demonstrates the basic feasibility of our approach.

3.7.2.1 MUC2 (Exploited by DroidCleaner) – Data in an email stored on the smartphone is stolen.

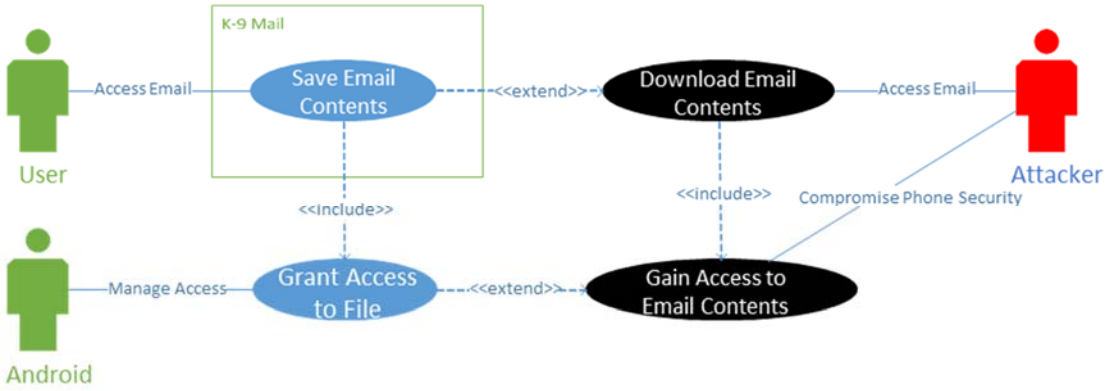

Figure 14: Misuse Case MUC2

In this misuse case, the user keeps email on the phone's External Storage area. The attacker gains access to the phone's storage by compromising the OS. A common way for the attacker to gain access to the phone is by tricking the user into installing a Trojan, to which the user unwittingly grants access to the drive during the install process. The attacker is then able to use the Trojan to download files, including the email contents file.

3.7.2.2 MUC4 (Exploited by DroidCleaner) – Confidential data in attachments is stolen.

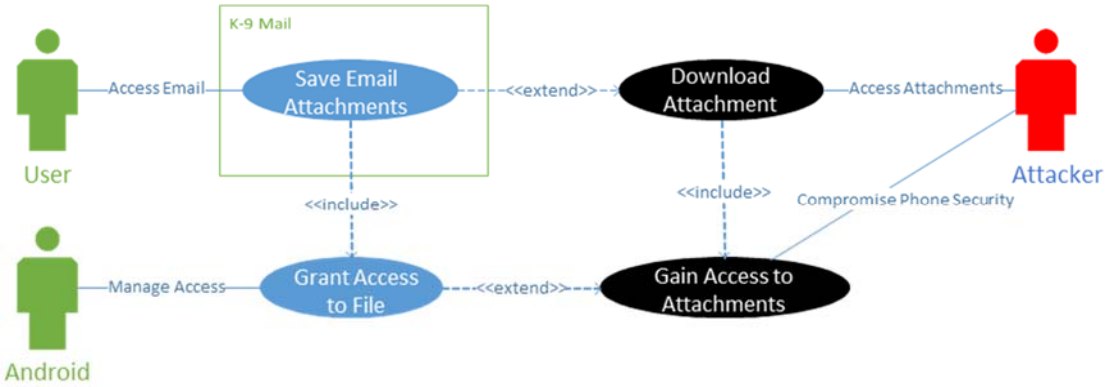

Figure 15: Misuse Case MUC4

In the *Confidential data in attachments is stolen* misuse case, the user stores attachments containing confidential information on the device's External Storage. In the *Gain access to attachments* misuse case, the attacker exploits the Android OS to gain access to the attachments in storage. This can be accomplished through a data-stealing attack. Once the attacker gains access to the attachments, the attacker uploads the attachments to a computer using the exploit on the Android device.

3.7.2.3 MUC1 – Email credentials are stolen.

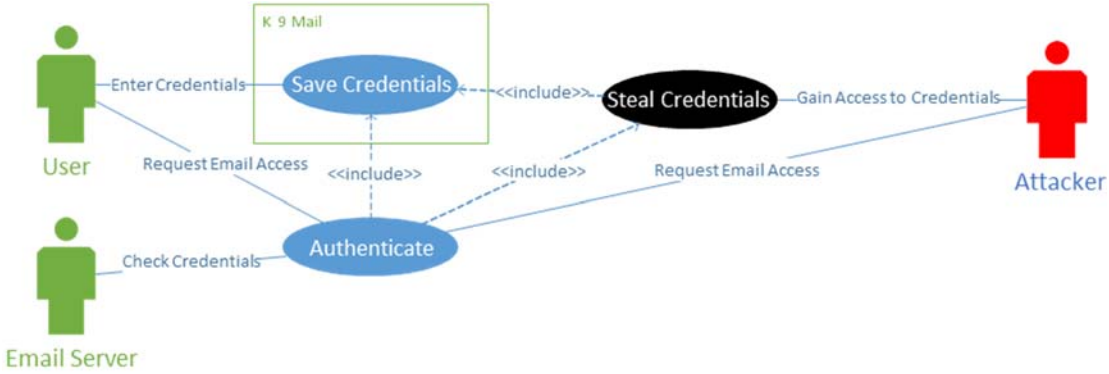

Figure 16: Misuse Case MUC1

In this misuse case, the user stores email credentials in the K-9 Mail application. An attacker then compromises the Android OS and steals the credential file. The attacker is then able to authenticate as the legitimate user to the email server by using the stolen credentials.

3.7.2.4 MUC3 – Data in an email in transit is stolen.

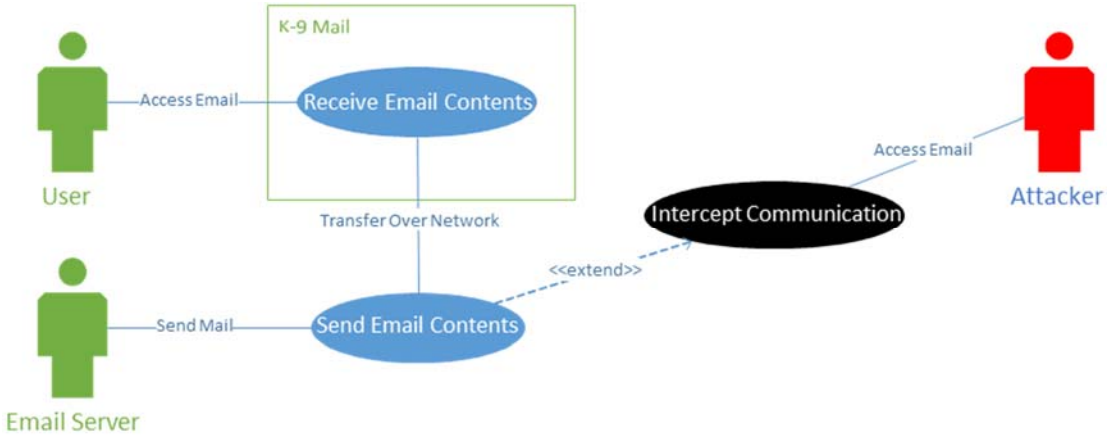

Figure 17: Misuse Case MUC3

In this misuse case, the attacker intercepts communication between the K-9 Mail client and the email server. To implement this misuse case, the attacker can set up a fake wireless access point that emulates a common unencrypted access point, such as the AT&T access points, Starbucks access points, or other common public access points. When the smartphone connects to the wireless network, the attacker is able to insert a computer as a node in the communication network to the internet and monitor data travelling between the K-9 client and the email server.

3.7.2.5 MUC5 – Attachment contains malware and is used to compromise the device.

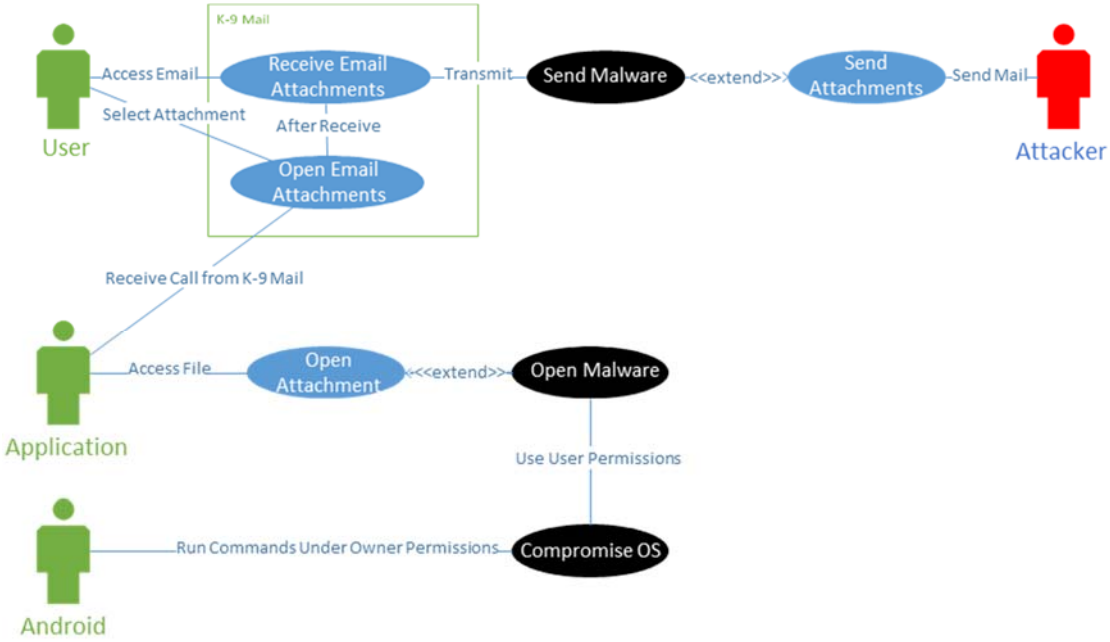

Figure 18: Misuse Case MUC5

In this misuse case, the K-9 Mail client is used as the attack vector for an exploit. The attacker writes an email and includes an attachment that contains malware. The user regularly receives email with attachments that are shown in the K-9 Mail client. The user then attempts to view the contents of attachments on the phone. The K-9 Mail client calls the native application for the attachment, which displays the contents of the attachment to the user. When the user opens the attachment, the K-9 Mail application will call the application specified by the MIME type. If an attacker sends malware and the user opens the malware, the attacker can compromise the operating system. The malware may be an installer package or a file containing an exploit for the application called by the file extension. Having gained control of the application, the attacker is able to perform actions on the OS with the permissions granted to the compromised application.

3.7.2.6 MUC6 – Emails are deleted without user's consent.

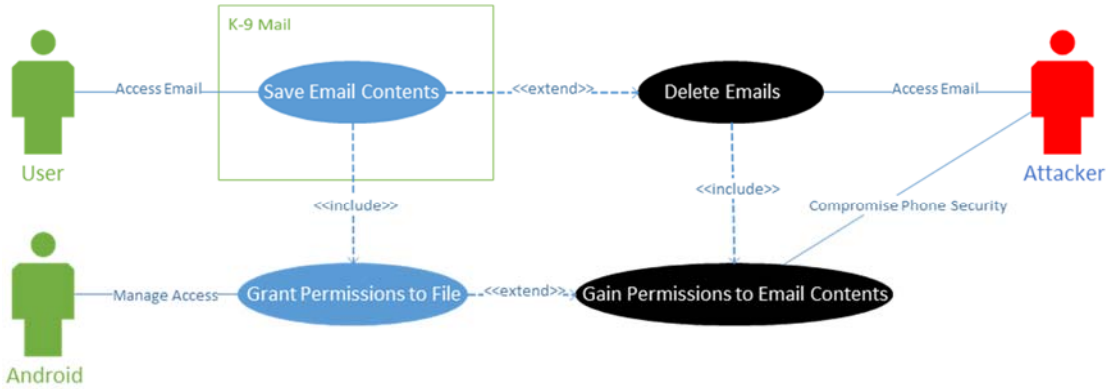

Figure 19: Misuse Case MUC6

In this misuse case, the user stores the email on the phone for convenient retrieval. If an attacker is able to trick the user into installing malware, the user will inadvertently grant the attacker permissions to delete files.

3.7.2.7 MUC7 – Attachments are deleted without user's consent.

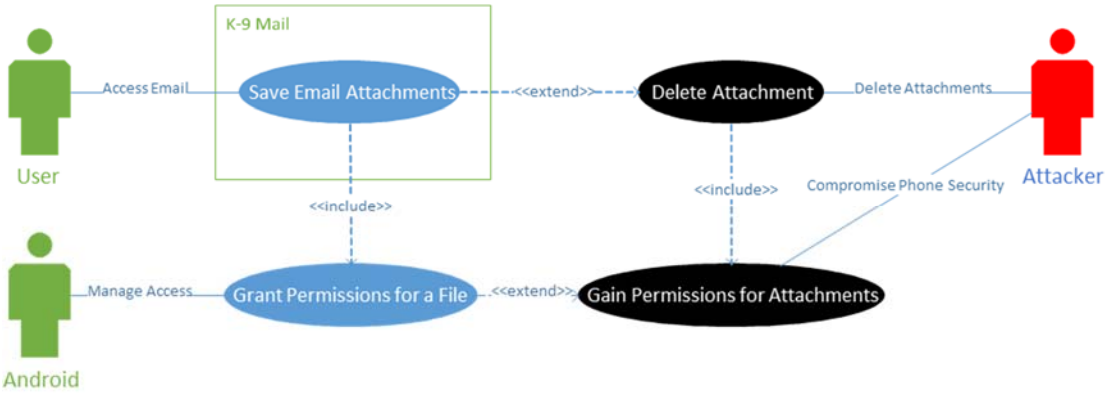

Figure 20: Misuse Case MUC7

In this misuse case, the user stores email attachments on the phone. The attacker then tricks the user into installing a Trojan, which is given permissions to delete files.

3.7.2.8 MUC8 – Emails are sent without user's consent.

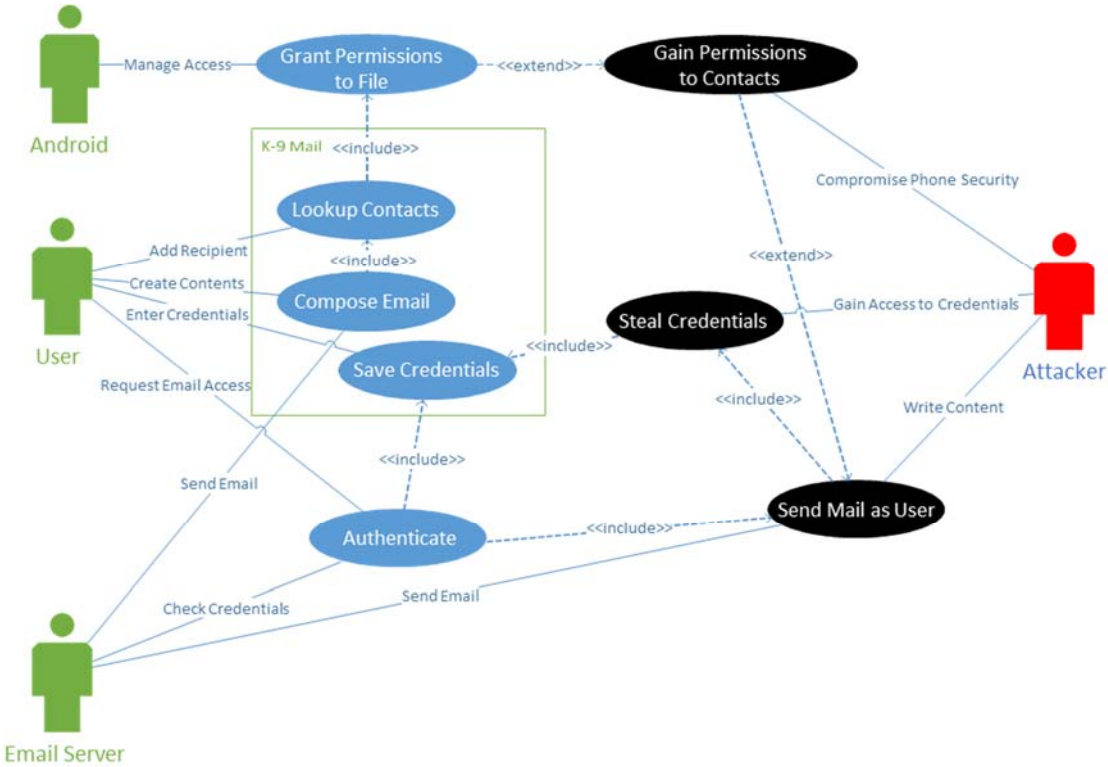

Figure 21: Misuse Case MUC8

In this misuse case, the attacker gains access to the credentials for the user's email account as well as the user's contacts. This misuse case can be performed through a Trojan or other attack, which

causes the Android OS to grant access to the file containing contacts. Having gained access to contacts and credentials, the attacker can send spoofed emails by submitting the stolen credentials to the email server. Once authenticated, the attacker is able to send emails using the user's account.

3.7.2.9 MUC9 – K-9 Mail application is placed on Google Play as a Trojan.

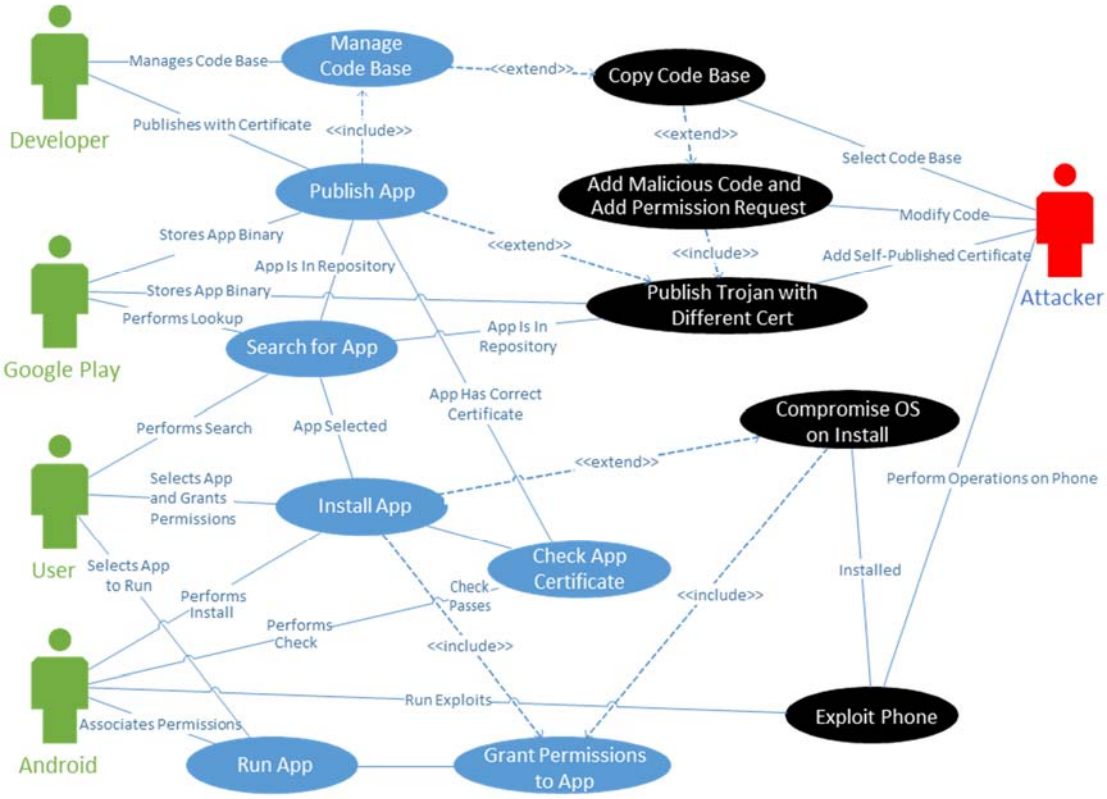

Figure 22: Misuse Case MUC9

In this misuse case diagram, the standard use cases for publishing, installing, and using the K-9 Mail application are shown in blue. The developer manages the code base. Once the application is complete, the developer promotes the application code to Google Play. A potential user will search the Google Play environment for applications meeting some criteria, and the user then selects the K-9 Mail application. The user then instructs Google Play to install K-9 Mail on the user's system. Then the Android OS reads the requested permissions from the install and presents the requested permissions to the user. If the user approves, then Google Play verifies the certificate used to sign the code is a valid certificate assigned to an individual or entity. If the certificate is valid, Android grants access to the services requested by the install manifest. The user may then start K-9 Mail. When this occurs, Android starts up the application in a sandbox with the permission set established in the install and then starts the application and presents it to the user.

In the misuse case, the attacker copies the source code repository. In the case of K-9 Mail, the application is open source, so the attacker simply requests the code as a volunteer. The attacker may modify the copy of the code base to include exploits, monitoring, or attacks against the OS. Then the attacker adds permission requests to the installation manifest, hoping the user will not carefully read the installation request from Google Play. After the code is complete, the attacker

compiles the code into a package that appears to be just like the original K-9 Mail application and signs the code with the attacker's own issued certificate. When the user installs the Trojan version of K-9 Mail, the user's OS is compromised. The Trojan uses the extra permission requests to give the attacker access to all storage and sensors on the device. When the user attempts to run the Trojan K-9 Mail, the extra code in the application works to break out of the application sandbox, and extra application code for stealing data and monitoring the user is able to run using the privileges granted during installation.

3.7.3 Detailed Requirement List

Table 6: Detailed Requirement List

Requirement Number: 1	
Requirement	1.1 Email contents shall be protected from unauthorized access. Email contents shall be stored in an area only available to the application (Android Internal Storage default configuration) and/or protected through encryption that cannot be decrypted using data available in Android External Storage. 1.2 Processes with access to External Storage shall not have the ability to view K-9 Mail contents in clear text. If External Storage is selected, a warning message or mitigation, such as encryption, is recommended.
Derived From	Initial Requirement 1, Misuse Case MUC2
Requirement Number: 2	
Requirement	1.1 Attachments shall be protected from unauthorized access. Attachments shall be stored in an area only available to the application (Android Internal Storage default configuration) and/or protected through encryption that cannot be decrypted using data available in Android External Storage. 1.2 Processes with access to External Storage shall not have the ability to view K-9 Mail attachments in clear text. If External Storage is selected, a warning message or mitigation, such as encryption, is recommended.
Derived From	Initial Requirement 2, Misuse Case MUC4
Requirement Number: 3	
Requirement	K-9 Mail shall not request access to SMS functionality on installation.
Derived From	Initial Requirement 3, Misuse Case MUC9
Requirement Number: 4	
Requirement	K-9 Mail shall verify the authenticity of the installation through a channel outside Google Play.
Derived From	Initial Requirement 3, Misuse Case MUC9
Requirement Number: 5	
Requirement	K-9 Mail shall provide a mechanism to scan attachments for malware prior to opening the attachment in an application. Providing interfaces for Android antivirus software to implement attachment-scanning functionality is recommended.
Derived From	Initial Requirement 4, Misuse Case MUC5
Requirement Number: 6	
Requirement	Email credentials shall not be revealed to anyone or any other process on the device. Credentials shall be stored in a manner that cannot be compromised by access to Android Internal Storage or Android External Storage.
Derived From	Initial Requirement 5, Misuse Cases MUC1, MUC8
Requirement Number: 7	
Requirement	Data and credentials transferred between the K-9 Mail application and all email servers shall be encrypted. Encryption will protect data transferred over networks from unauthorized disclosure.

Derived From	Initial Requirement 6, Misuse Case MUC3
Requirement Number: 8	
Requirement	K-9 Mail shall open attachments by calling the application associated to the specific MIME type of the attachment in read-only mode. Call the application, pass in the specific MIME type, and specify the ACTION_VIEW intent. Using the specific MIME type and read-only mode will reduce the odds that an executable attachment can be disguised as a data file and opened as an executable.
Derived From	Initial Requirement 4, Misuse Case MUC5
Requirement Number: 9	
Requirement	K-9 Mail shall ensure that links in email are opened intentionally.
Derived From	Initial Requirement 4, Misuse Case MUC5
Requirement Number: 10	
Requirement	K-9 Mail shall store email credentials in a location in which the OS allows access only by the application (Android Internal Storage), and the credentials shall be encrypted. Storing credentials in a protected area will prevent Trojans from capturing the credentials.
Derived From	Initial Requirement 5, Misuse Case MUC1

3.8 SQUARE Step 7: Categorize Requirements

The following requirements categories were identified for the K-9 Mail application based on the security requirements:

- Client-Server Interaction
- Data Protection
- OS Protection

Table 7: Categorized Security Requirements List for K-9 Mail

Requirement Number: 1	
Requirement	1.1 Email contents shall be protected from unauthorized access. Email contents shall be stored in an area only available to the application (Android Internal Storage default configuration) and/or protected through encryption that cannot be decrypted using data available in Android External Storage. 1.2 Processes with access to External Storage shall not have the ability to view K-9 Mail contents in clear text. If External Storage is selected, a warning message or mitigation, such as encryption, is recommended.
Category	Data Protection
Requirement Number: 2	
Requirement	1.1 Attachments shall be protected from unauthorized access. Attachments shall be stored in an area only available to the application (Android Internal Storage default configuration) and/or protected through encryption that cannot be decrypted using data available in Android External Storage. 1.2 Processes with access to External Storage shall not have the ability to view K-9 Mail attachments in clear text. If External Storage is selected, a warning message or mitigation, such as encryption, is recommended.
Category	OS Protection
Requirement Number: 3	
Requirement	K-9 Mail shall not request access to SMS functionality on installation.
Category	OS Protection
Requirement Number: 4	
Requirement	K-9 Mail shall verify the authenticity of the installation through a channel outside Google Play.

Category	OS Protection
Requirement Number: 5	
Requirement	K-9 Mail shall provide a mechanism to scan attachments for malware prior to opening the attachment in an application. Providing interfaces for Android antivirus software to implement attachment-scanning functionality is recommended.
Category	OS Protection
Requirement Number: 6	
Requirement	Email credentials shall not be revealed to anyone or any other process on the device. Credentials shall be stored in a manner that cannot be compromised by access to Android Internal Storage or Android External Storage.
Category	Data Protection
Requirement Number: 7	
Requirement	Data and credentials transferred between the K-9 Mail application and all email servers shall be encrypted. Encryption will protect data transferred over networks from unauthorized disclosure.
Category	Client-Server Interaction
Requirement Number: 8	
Requirement	K-9 Mail shall open attachments by calling the application associated to the specific MIME type of the attachment in read-only mode. Call the application, pass in the specific MIME type, and specify the ACTION_VIEW intent. Using the specific MIME type and read-only mode will reduce the odds that an executable attachment can be disguised as a data file and opened as an executable.
Category	OS Protection
Requirement Number: 9	
Requirement	K-9 Mail shall ensure that links in email are opened intentionally.
Category	OS Protection
Requirement Number: 10	
Requirement	K-9 Mail shall store email credentials in a location in which the OS allows access only by the application (Android Internal Storage), and the credentials shall be encrypted. Storing credentials in a protected area will prevent Trojans from capturing the credentials.
Category	Data Protection

3.9 SQUARE Step 8: Prioritize Requirements

Based on the analysis in the previous sections, including risk analysis, misuse cases, and attack trees, the following security requirements apply to the K-9 Mail application. The cost values are based on prior experience rather than on the complexity of including the functionality in the code. The requirements for K-9 Mail were prioritized based on the risk level and complexity (cost of implementation).

Table 8: K-9 Mail Final Security Requirements

Requirement Number: 1	
Requirement	1.1 Email contents shall be protected from unauthorized access. Email contents shall be stored in an area only available to the application (Android Internal Storage default configuration) and/or protected through encryption that cannot be decrypted using data available in Android External Storage. 1.2 Processes with access to External Storage shall not have the ability to view K-9 Mail contents in clear text. If External Storage is selected, a warning message or mitigation, such as encryption, is recommended.
Category	Data Protection

Priority	High
Cost	Medium
Misuse Case	MUC2
Rationale	Due to the high risk of data-theft malware on Android, it is not safe to assume data kept on the phone is private, so the email contents must be kept in a form that cannot be read even if the attacker has access to the storage location.
Requirement Number: 2	
Requirement	1.1 Attachments shall be protected from unauthorized access. Attachments shall be stored in an area only available to the application (Android Internal Storage default configuration) and/or protected through encryption that cannot be decrypted using data available in Android External Storage. 1.2 Processes with access to External Storage shall not have the ability to view K-9 Mail attachments in clear text. If External Storage is selected, a warning message or mitigation, such as encryption, is recommended.
Category	OS Protection
Priority	High
Cost	Medium
Misuse Case	MUC4
Rationale	Due to the high risk of data-theft malware on Android, it is not safe to assume data kept on the phone is private, so the attachments must be kept in a form that cannot be read even if the attacker has access to the storage location.
Requirement Number: 3	
Requirement	K-9 Mail shall not request access to SMS functionality.
Category	OS Protection
Priority	High
Cost	Low
Misuse Case	MUC9
Rationale	A Trojan will appear much more suspicious if it is requesting SMS messaging at installation. SMS messaging is the most common attack vector for Android malware, and K-9 Mail does not provide SMS functionality. Following the principles of least privilege, K-9 Mail should not access SMS messaging.
Requirement Number: 4	
Requirement	K-9 Mail shall verify the authenticity of the installation through a channel outside Google Play.
Category	OS Protection
Priority	High
Cost	Medium
Misuse Case	MUC9
Rationale	A common attack vector for Android devices is the use of Trojan applications, which are a copy of legitimate applications with extra code to exploit Android. K-9 Mail should provide additional verification beyond the Google Play store to ensure no Trojans of K-9 Mail exist.
Requirement Number: 5	
Requirement	K-9 Mail shall provide a mechanism to scan attachments for malware prior to opening the attachment in an application. Providing interfaces for Android antivirus software to implement attachment-scanning functionality is recommended.
Category	OS Protection
Priority	Medium
Cost	High
Misuse Case	MUC5
Rationale	In the future, Android malware may be spread through email.

Requirement Number: 6	
Requirement	Email credentials shall not be revealed to anyone or any other process on the device. Credentials shall be stored in a manner that cannot be compromised by access to Android Internal Storage or Android External Storage.
Category	Data Protection
Priority	High
Cost	Medium
Misuse Case	MUC1, MUC8
Rationale	Due to the high risk of data-theft malware on Android, it is not safe to assume data kept on the phone is private, so the credential storage must be kept in a form that cannot be read even if the attacker has access to the storage location.
Requirement Number: 7	
Requirement	Data and credentials transferred between the K-9 Mail application and all email servers shall be encrypted. Encryption will protect data transferred over networks from unauthorized disclosure.
Category	Client-Server Interaction
Priority	Medium
Cost	Medium
Misuse Case	MUC3
Rationale	The internet is not a secure transfer protocol, and users have the expectation of privacy for the contents of email.
Requirement Number: 8	
Requirement	K-9 Mail shall open attachments by calling the application associated to the specific MIME type of the attachment in read-only mode. Call the application, pass in the specific MIME type, and specify the ACTION_VIEW intent. Using the specific MIME type and read-only mode will reduce the odds that an executable attachment can be disguised as a data file and opened as an executable.
Category	OS Protection
Priority	High
Cost	Medium
Misuse Case	MUC5
Rationale	In the future, Android malware may be spread through email.
Requirement Number: 9	
Requirement	K-9 Mail shall ensure that links in email are opened intentionally.
Category	OS Protection
Priority	Medium
Cost	Low
Misuse Case	MUC5
Rationale	This setting will prevent users from opening links and attachments that are potentially harmful. Providing granularity will prevent the message from becoming too annoying. A recommendation for implementation is to provide a setting to confirm opening attachments and links with three levels: Always Not Contacts – This will display a warning when opening a link or attachment for emails from addresses that are not contacts. Never (not recommended)
Requirement Number: 10	
Requirement	K-9 Mail shall store email credentials in a location in which the OS allows access only by the application (Android Internal Storage), and the credentials shall be encrypted.
Category	Data Protection
Priority	High

Cost	Medium
Misuse Case	MUC1
Rationale	Storing credentials in a protected area will prevent Trojans from capturing the credentials.

3.10 SQUARE Step 9: Inspect Requirements

3.10.1 Requirements Review

The requirements were reviewed by subject matter experts at the Software Engineering Institute and the School of Computer Science at Carnegie Mellon University. Comments were collected via email and addressed. Further validation of the security requirements was performed by tracing the requirements to the attack tree.

3.10.2 Attack Tree Trace

The attack tree that covers the security flaw for DroidCleaner was analyzed using the developed set of security requirements to ensure the requirements block all paths through the attack tree. Red octagons indicate the points in the attack tree that are mitigated by a security requirement. In the attack tree shown in Figure 23, all paths through the attack tree are mitigated by a security requirement, which is a strong indication that the set of security requirements will block the DroidCleaner malware.

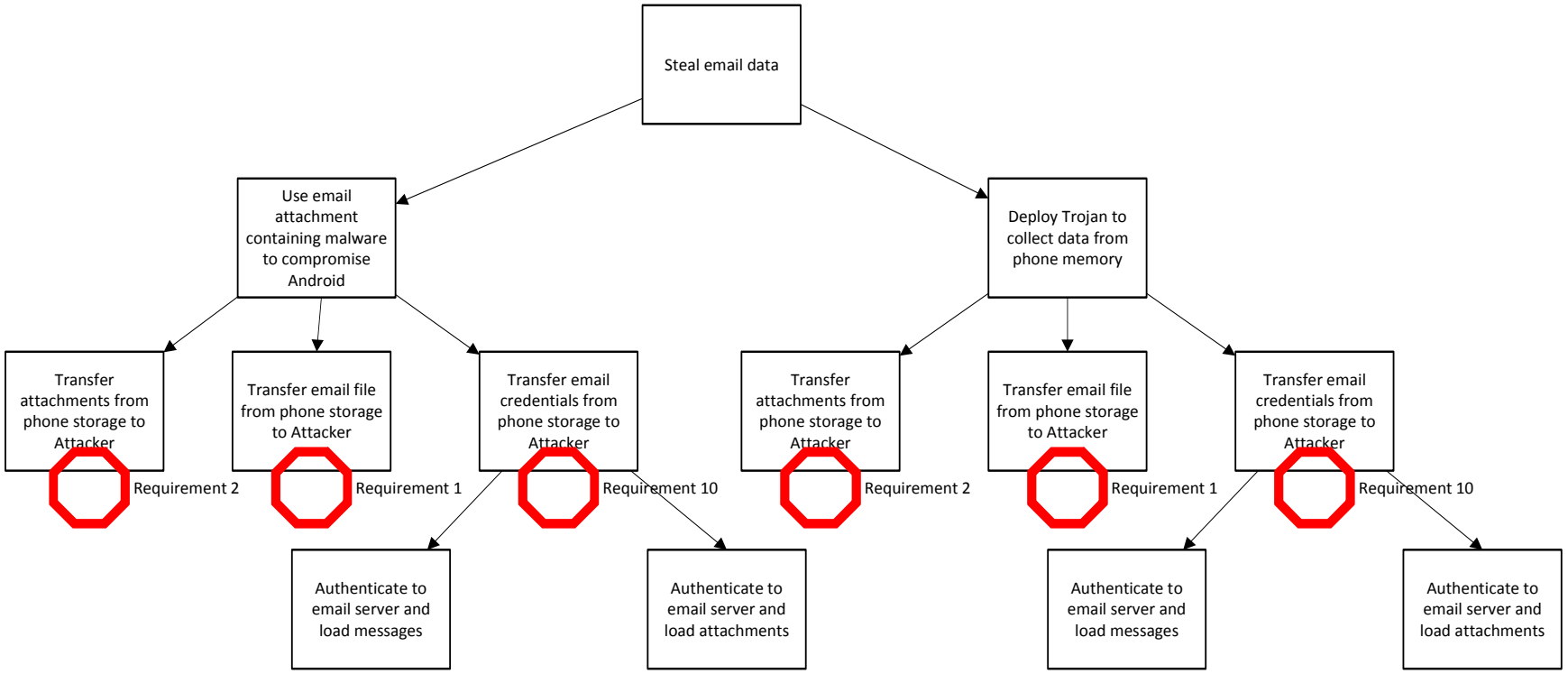

Figure 23: Mapping of DroidCleaner Attack Tree to Requirements

4 Evaluation

4.1 Malware Coverage

Adding Android malware analysis to the SQUARE process when defining requirements on the K-9 Mail application emphasizes the data protection requirements. These requirements were investigated more fully as a result of having identified the malware. Malware analysis was useful in SQUARE's Risk Assessment step and provided additional data to enhance risk analysis based on security threats encountered in the field. Analyzing the malware classes for the application platform guided the definition of security requirements. Examination of platform malware provided realistic data for determining the likelihood of occurrence for given security risks.

At the start of the project, the theft of credentials appeared to be the greatest threat to the application, based on experience. Malware analysis demonstrated that Trojanization of the application is a significant threat. Furthermore, malware analysis demonstrated that even though credentials are a valuable target, malware has not yet targeted them. Data-stealing malware, in practice, targets the open permissions design of Android's External Storage area, so the lack of protection for the email contents in storage is, in practice, a greater threat to the confidentiality of email than credentials.

Malware analysis also provided valuable input for the Requirements Prioritization step of SQUARE. Rather than prioritizing requirements based on intuitive judgments, prioritization decisions were made based on quantifiable data from analysis of malware in the field.

Malware analysis integrates well with SQUARE's third step, "Develop artifacts to support security requirements definition," because malware analysis can be an artifact for requirements definition. Misuse cases can be derived from targeted malware or malware that could potentially compromise a security goal. Malware analysis effectively provided data points to accurately assess risk in SQUARE's step 4, "Perform Risk Assessment."

Currently, the lack of tool support for malware analysis makes it difficult to use in the requirements definition process. In this case study, the malware that can threaten K-9 Mail with data stealing (DroidCleaner) was discovered through a prior mobile malware survey paper and numerous Google searches for different criteria. Not all malware resources that discussed DroidCleaner mentioned the data upload functionality. Malware analysis for SQUARE could be improved by creating a common database of malware with application platforms, application data writing, data storage locations, communication protocol use, and malware behaviors. Such a searchable database would make identifying potential malware much more productive for security requirements engineers.

4.2 Mobile Application Development

The SQUARE process is effective for developing security requirements for mobile applications. Mobile applications generally have a smaller feature set than general computing applications and are generally installed on devices with a single user. As a result, mobile applications tend to have fewer security requirements than general computing applications. As a result, in this case study,

SQUARE's Step 1, *Agree on definitions*, Step 4, *Perform risk assessment*, and Step 7, *Categorize requirements as to level (system, software, etc.) and whether they are requirements or other kinds of constraints*, were abbreviated compared to the usage necessary in large-scale computing applications.

5 Conclusions

The security requirements for the K-9 Mail application are a combination of the expected value of the contained data and the user's expectations of privacy of the data handled by the application, in combination with the security risks inherent to the computing platform.

At the beginning of the risk analysis phase, the security risks of data theft and a loss of confidentiality of email contents was expected to be the greatest risk to the K-9 application. Based on analysis of existing malware for the Android platform, the greatest risk, based on frequency of occurrence, is the possibility of an attacker "Trojanizing" the application by creating a version of K-9 Mail with extra code to exploit the privileges given by the user on install. This risk unexpectedly generated the highest CVSS score and, as a result, security requirements to prevent the application from being manipulated into a Trojan became important requirements to develop. Commercial applications with proprietary source code have been reverse-engineered into Trojans, so the challenge of preventing an open source application from becoming a widely distributed Trojan is significant.

The method selected for preventing the proliferation of Trojan versions of K-9 Mail involves implementing multiple checks for authenticity so there is a greater chance the attacker will miss a check when modifying the code, arousing user suspicion. The user could also check the authenticity of the user's installation from the official K-9 Mail website. Using the help website will bypass the Google Play and application instance for verifying the authenticity of the application. This external channel will provide an opportunity to check the authenticity of the distribution outside of the attacker's control. The remaining security requirements are based on the premise that the Android platform on which the software is operating will be compromised, making encryption and other secure techniques necessary to prevent disclosure of the data if the data is copied from the device by a Trojan.

Mobile computing applications have a more limited feature set than large-scale computing applications and, as a result, have fewer security requirements, which causes some of the steps in SQUARE that focus on organizing security requirements to be simplified to the point of triviality.

Malware analysis integrates well with SQUARE by providing scenarios that yield misuse case artifacts to support the development of security requirements and provide data on which risk analysis can be conducted. Malware analysis requires tools to support its use in an industrial setting. Such tools would simplify data collection for building security requirements for any computing system platform.

SQUARE can be enhanced by enhancing Step 3 of the SQUARE framework, "Develop Artifacts to Support Requirements Elicitation." Adopting a method of analyzing malware to identify missed security requirements for current, in-production software will provide an opportunity to incorporate these requirements into new software. Furthermore, completing malware analysis prior to risk analysis will provide additional data for the risk analysis process. If attack trees are developed for malware that poses a risk to software in development, the requirements developed using the SQUARE methodology can be validated by ensuring that each branch of the attack tree is mitigated by a security requirement.

Malware analysis, with some additional tool development, can provide a significant enhancement to the development of security requirements by providing a feedback loop to ensure that system security evolves with the increasing sophistication of malware. Unlike functional requirements, security requirements have a shelf life. A malware analysis feedback loop in a security requirements definition process should occur regularly to ensure that if malware advancements necessitate updates to production software, these requirements are identified and incorporated proactively prior to an exploit.

References

URLs are valid as of the publication date of this document.

[Ahmad 2013]
Ahmad, M.; Musa, N.; Nadarajah, R.; Hassan, R.; & Othman, N. "Comparison Between Android and iOS OS in Terms of Security," 1-4. *2013 8th International Conference on Information Technology in Asia (CITA)*. Bangi, Selangor, Malaysia, July 2013. IEEE, 2013.

[Balanza 2011]
Balanza, M.; Alintanahin, K.; Abendan, O.; Dizon, J.; & Caraig, B. "DroidDreamLight Lurks Behind Legitimate Android Apps," 73-78. *2011 6th International Conference on Malicious and Unwanted Software (MALWARE)*. Fajardo, Puerto Rico, USA. Oct. 2011. IEEE, 2011.

[Chebyshev 2013]
Chebyshev, V. *Mobile Attacks!* http://securelist.com/blog/virus-watch/65379/mobile-attacks/ (2013).

[Detica 2011]
Detica Limited. *The Cost of Cyber Crime*. Detica Limited, 2011.

[Dong 2014]
Dong, Q.; Zhao, R.; & Sun, N. *Oldboot.B: The Hiding Tricks Used by bootkit on Android*. http://blogs.360.cn/360mobile/2014/04/02/analysis_of_oldboot_b_en/ (2014).

[Elahi 2010]
Elahi, G.; Yu, E.; & Zannone, N. "A Vulnerability-Centric Requirements Engineering Framework: Analyzing Security Attacks, Countermeasures, and Requirements Based on Vulnerabilities." *Requirements Engineering 15*, 1 (March 2010): 41-62.

[F-Secure 2014]
F-Secure. *Mobile Threat Report Q1 2014*. http://www.f-secure.com/documents/996508/1030743/Mobile_Threat_Report_Q1_2014.pdf (2014).

[Google 2014]
Google. *Android Security Overview*. https://source.android.com/devices/tech/security/ (2014).

[ISO 2008]
International Organization for Standardization (ISO). ISO/IEC 21827:2008, Information Technology -- Security Techniques -- Systems Security Engineering -- Capability Maturity Model® (SSE-CMM®). ISO, 2008.

[ISO 2009]
International Organization for Standardization (ISO). ISO/IEC 15408:2009, Information Technology -- Security Techniques -- Evaluation Criteria for IT Security. ISO, 2009.

[Jacobson 1992]
Jacobson, I. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, 1992.

[Jiang 2011]
Jiang, X. *Security Alert: New Sophisticated Android Malware DroidKungFu Found in Alternative Chinese App Markets*. http://www.cs.ncsu.edu/faculty/jiang/DroidKungFu/ (2011).

[K-9 2014]
K-9. *K-9 Mail Manual*. https://github.com/k9mail/k-9/wiki/Manual (2014).

[Khan 2012]
Khan, S.; Nauman, M.; Othman, A. T.; & Musa, S. "How Secure Is Your Smartphone: An Analysis of Smartphone Security Mechanisms," 76-81. *2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*. Kuala Lumpur, Malaysia, June 2012. IEEE, 2012.

[La Polla 2013]
La Polla, M.; Martinelli, F.; & Sgandurra, D. "A Survey on Security for Mobile Devices." *IEEE Communications Surveys & Tutorials 15*, 1 (First Quarter 2013): 446-471.

[Lookout 2012]
Lookout. *Lookout's Take on the 'Apperhand' SDK (aka 'Android.Counterclank')*. https://blog.lookout.com/blog/2012/01/27/lookout%E2%80%99s-take-on-the-%E2%80%98apperhand%E2%80%99-sdk-aka-android-counterclank/ (2012).

[Mead 2005]
Mead, N. R.; Hough, E. D.; & Stehney II, T. R. *Security Quality Requirements Engineering (SQUARE) Methodology* (CMU/SEI-2005-TR-009). Software Engineering Institute, Carnegie Mellon University, 2005. http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=7657

[Mead 2014]
Mead, N. R. & Morales, J. A. "Using Malware Analysis to Improve Security Requirements on Future Systems," 37-41. *2014 IEEE 1st International Workshop on Evolving Security and Privacy Requirements Engineering (ESPRE)*. Karlskrona, Sweden, Aug. 2014. IEEE, 2014.

[Mellado 2007]
Mellado, D.; Fernandez-Medina, E.; & Piattini, M. "A Common Criteria Based Security Requirements Engineering Process for the Development of Secure Information Systems." *Computer Standards & Interfaces 29*, 2 (February 2007): 244-253.

[Microsoft 2011]
Microsoft. *Trojan:AndroidOS/Fakeplayer.A*. Microsoft, 2011. http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Trojan:AndroidOS/Fakeplayer.A

[Microsoft 2014]
Microsoft. *What Is Spyware?* Microsoft, 2014. http://www.microsoft.com/security/pc-security/spyware-whatis.aspx

[MITRE 2014]
MITRE. *Common Weakness Enumeration*. http://cwe.mitre.org/ (2014).

[Mundie 2013]
Mundie, D. & McIntire, D. "An Ontology for Malware Analysis," 556-558. *2013 International Conference on Availability, Reliability and Security, ARES 2013.* Regensburg, Germany, Sep. 2013. IEEE Computer Society, 2013.

[NIST 2014]
National Institute of Standards and Technology (NIST). *Common Vulnerability Scoring System Version 2 Calculator*. http://nvd.nist.gov/cvss.cfm?calculator&version=2 (2014).

[Paoli 2013]
Paoli, C. New Android Malware Aims To Infect PCs. http://redmondmag.com/articles/2013/02/06/android-malware-aims-to-infect-pc.aspx (2013).

[Paturi 2013]
Paturi, A.; Cherukuri, M.; Donahue, J.; & Mukkamala, S. "Mobile Malware Visual Analytics and Similarities of Attack Toolkits," 149-154. *2013 International Conference on Collaboration Technologies and Systems (CTS)*. San Diego, CA, May 2013. IEEE, 2013.

[Peng 2014]
Peng, S.; Yu, S.; & Yang, A. "Smartphone Malware and Its Propagation Modeling: A Survey." *IEEE Communications Surveys & Tutorials 16*, 2 (Second Quarter 2014): 925-941.

[Salechie 2012]
Salechie, M.; Pasquale, L.; Omoronyia, I.; Ali, R.; & Nuseibeh, B. "Requirements-Driven Adaptive Security: Protecting Variable Assets at Runtime," 111-120. *Requirements Engineering Conference (RE), 2012 20th IEEE International, Proceedings.* Chicago, IL, Sep. 2012. IEEE, 2012.

[Sophos 2014]
Sophos Ltd, "Sophos Mobile Security Threat Report," SophosLabs, Oxford, UK, and Boston, USA, 2014.

[Symantec 2010]
Symantec. *Security Response: Android.Tapsnake*. http://www.symantec.com/security_response/writeup.jsp?docid=2010-081214-2657-99 (2010).

[Symantec 2012]
Symantec. *Security Response: Android.Counterclank.* http://www.symantec.com/security_response/writeup.jsp?docid=2012-012709-4046-99 (2012).

[Unuchek 2014]
Unuchek, R. *The First Tor Trojan for Android*. http://securelist.com/blog/incidents/58528/the-first-tor-trojan-for-android/ (2014).

[Walker 2013]
Walker, D. “Android App Lies to Users that Their Device Is Infected by Viruses, Asks for Money.” *SC Magazine* (June 21, 2013).

[Yin 2012]
Yin, S. “Chinese Espionage Campaign ‘Luckycat’ Targets Android.” *PCMag* (July 31, 2012).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	November 2014	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Using Malware Analysis to Tailor SQUARE for Mobile Platforms	FA8721-05-C-0003

6. AUTHOR(S)

Gregory Paul Alice, Nancy R. Mead (Faculty Adviser)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2014-TN-018

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFLCMC/PZE/Hanscom Enterprise Acquisition Division 20 Schilling Circle Building 1305 Hanscom AFB, MA 01731-2116	n/a

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

As the number of mobile-device software applications has grown, so has the amount of malware targeting them. More than 650,000 pieces of malware now target the Android platform. As mobile malware becomes more sophisticated and begins to approach threat levels seen on PC platforms, software development security practices for mobile applications will need to adopt the security practices for PC applications to reduce consumers' exposure to financial and privacy breaches on mobile platforms. This technical note explores the development of security requirements for the K-9 Mail application, an open source email client for the Android operating system. The project's case study (1) used the Security Quality Requirements Engineering (SQUARE) methodology to develop K-9 Mail's security requirements and (2) used malware analysis to identify new security requirements in a proposed extension to the SQUARE process. This second task analyzed the impacts of DroidCleaner, a piece of Android malware, on the security goals of the K-9 Mail application. Based on the findings, new requirements are created to ensure that similar malware cannot compromise the privacy and confidentiality of email contents.

14. SUBJECT TERMS	15. NUMBER OF PAGES
malware analysis, SQUARE, mobile malware, security requirements, Android security	61

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL